Z80ファミリ・ハンドブック

Z80 CPU・PIO・CTC・DMA・SIOの使い方のすべて

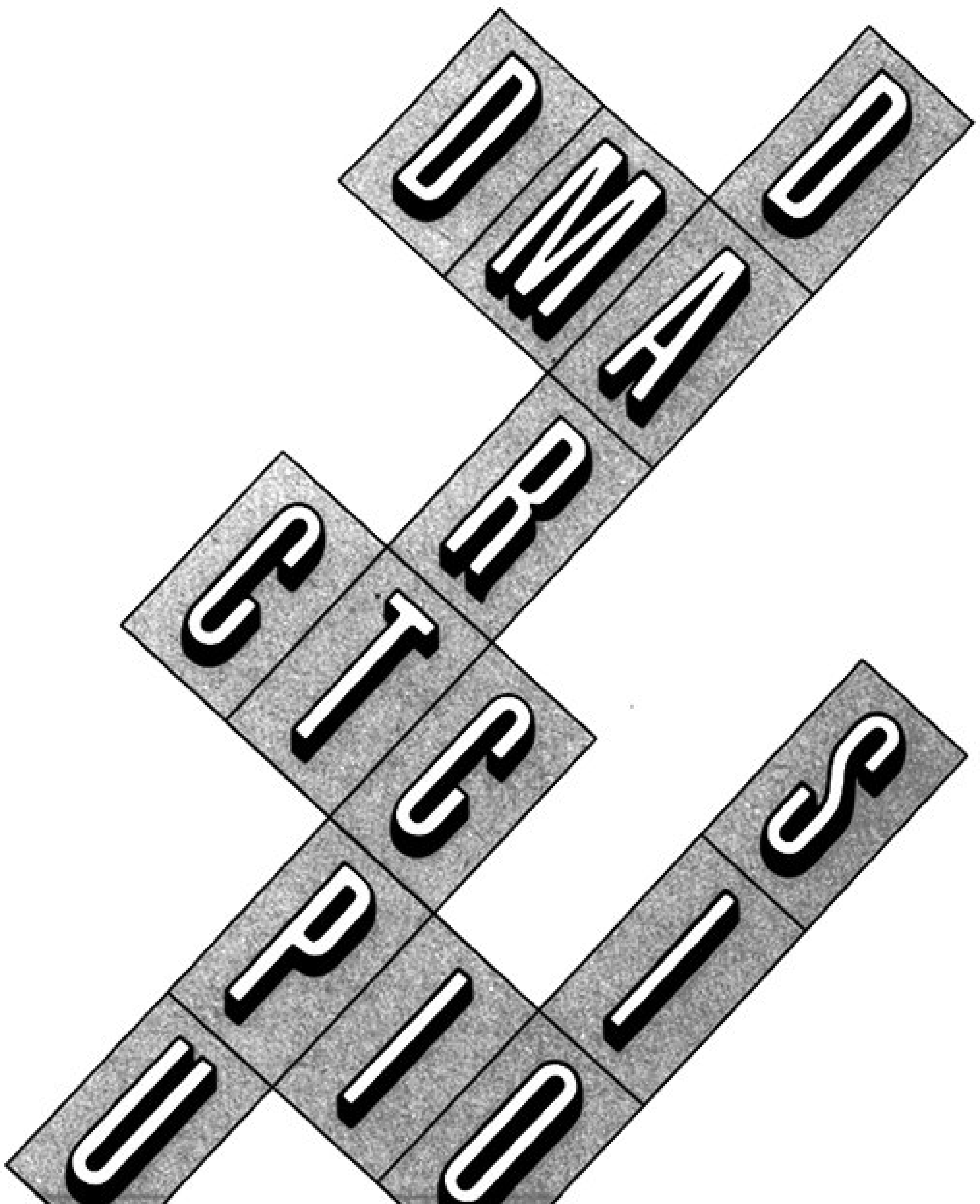
額田忠之 著



Z80ファミリ・ハンドブック

Z80 CPU・PIO・CTC・DMA・SIOの使い方のすべて

額田忠之 著



CQ出版社

まえがき

Z80 CPUが世に現われてからはや10年にならんとしていますが，技術革新，世代交代の早いこのエレクトロニクス界にあって，実に10年の間マイクロプロセッサの王者としてZ80が君臨し，今後もその座を維持し続けるであろうことは驚異に値することではないでしょうか．

Z80 CPUが長期間このような地位を占めてこられた理由はいろいろありますが，第一にいえることはZ80 CPUが優秀な性能をもった周辺LSI(PIO, CTC, SIO, DMA)により，がっちりと身を守り，Z80ファミリを構成したことです．これらの周辺LSIの機能は，現時点においても最高のレベルにあり，かつZ80 CPUと一緒に使うことにより，フル機能を発揮できるように設計されています．

世の中は８ビットCPUの時代から16ビット，または32ビットCPUの時代に移行しつつあるように見えます．この理由は，マイクロプロセッサの利用方法が高度化し，８ビットCPUではパーフォーマンス的に不十分であることもありますが，一番大きな理由はダイレクト・アドレッシングできるメモリの容量にあります．

ご存知のように最近のメモリの低価格化には目を見はるものがあり，マイクロプロセッサの下に１Mバイトくらいのメモリを置く応用もまれではなくなっています．

ところが，Z80などで代表される8ビットCPUによりダイレクト・アドレッシングできるメモリの容量は64Kバイトであり，それ以上のメモリ容量を制御しようとすると，複雑なバンク構造をとらなければならず，それを制御するソフトウエアに対して大きな負担と困難を強いるだけでなく，パーフォーマンス的にも犠牲を覚悟しなければなりません．

したがって，大容量のメモリを容易に制御できる16ビット以上のCPUの利用が今後ますます広がっていくことは誰もが認めるところですが，それではZ80で代表される８ビットCPUは不要になるか? というとその答は“ノー”であり，今後もZ80 CPUおよびそのファミリは使われ続けていくことでしょう．あまり大きなメモリを必要としない応用，たとえば比較的単純なコントロール・アプリケーションなどでは，64Kバイトもメモリがあれば十分です．

また16ビット，32ビットCPUの世界では，CPUの価格そのものは８ビットCPUに比べて必然的に高くなります．したがって８ビットCPUで十分できることを16ビット以上のCPUでやらせることは非常にムダであるということができます．たとえばI/Oオペレーションなどがあげられます．

16ビット以上のCPUの応用においても，キーボードやプリンタ，フロッピー・ディスク，さらにはハード・ディスクなどのI/Oが使われますが，これらのI/Oの制御に高価なCPUを使うことには，大きなムダがあります．

したがって単純なI/Oオペレーションなどはすべて８ビットCPUに委せ，高価ではあるが高性能な16ビット以上のCPUには算術演算などの，より複雑な仕事に専念させるという機能分散が進んでいくに違いありません．

16ビット，または32ビットCPUバスの下に，複数のI/Oコントロール・モジュールがぶら下がり，各I/Oコントロール・モジュールにはそれぞれZ80 CPUと周辺LSIが使われ，キーボード，フロッピー・ディスクやハード・ディスクなどのI/Oがぶら下がります．

また，シリアル・インターフェース用のI/Oコントロール・モジュールを考えることもでき，これはリンク・レベルのプロトコルを制御し，上位のCPUとはデータのやりとりだけをします．つまり，雑用はすべての８ビットCPUに委せ，高価なCPUは本業に専念できます．

このような夢(実際には夢ではなくこのような使いかたはよくされている)が実現可能な理由の一つには，Z80ファミリのICが非常に低価格で入手できることがあげられます．事実，秋葉原などではZ80A CPUは数100円で売られています．あれだけ高機能を持ったCPUがその程度の値段で買えるのですから，今後も目的に応じてZ80およびそのファミリを使用していく必要がありますし，かつ使用せざるをえないケースに直面することでしょう．

このように「Z80は永遠である」という観点から，本書ではZ80 CPU，およびその周辺LSIについて詳しく説明をしてあります．今後Z80を学ぶ人にとっては教科書として，またすでに使用されている人にとってはハンド・ブックのようなものとして使われることを期待しています．

1984年11月

著　者

Z80ファミリ・ハンドブック

CONTENTS

●デザイン　辛　良二
●編集担当　浜田　雅生

CONTENTS

第1章

Z80 CPU編

Z80 CPUの特徴

Z80 CPU(以下Z80)は発表当時(1976年)，マイクロプロセッサの世界を牛耳っていたインテル社の8080Aを駆逐すべく，かつ8080Aのユーザーをも奪いとってしまうことを目的として開発されたCPUであり，その設計理念は「8080Aとの互換性を保ちつつ，よりパフォーマンスの高いCPUを作る」ことにありました．したがってZ80の特徴を述べる際には，いやでも8080Aを引き合いに出さなくてはなりません．

Z80と8080Aとの比較を行うと，読者は“8080AはまるでダメなCPU”という印象を受けられると思いますが，8080Aは人類が始めて創り出した(1975年)本格的なマイクロプロセッサであり，依然としてこの世界における“星”であることをお忘れなく．そしてZ80は8080Aの改良版である，ということを常に念頭においてください．

以下にZ80の特徴を列挙し，個々に説明を加えていきます．

❶ 8080Aのソフトウェアと完全な互換性がある

何といってもZ80最大の特徴は，8080Aのソフトウェアと完全な互換性があるということです．つまり，8080A用に作られたプログラムが，何の変更を加える必要もなく，Z80上で動作することができることです．

Z80では，8080Aが持つ命令をすべて含み，かつその上に有用な命令を付け加えていったからです．もしZ80にこの特徴がなかったとしたら，Z80が現在ある地位にいることができたかどうかは，大いに疑問です．

❷ 8080Aにはない，有用な命令が多くある

命令の種類の数えかたは人によって多少異なりますが，一般的には8080Aの持つ命令の種類は78種，Z80のそれは158種となっています．つまり，Z80は8080Aの約2倍の命令数があるわけです．Z80で増設された命令のうち，とくに有用なものは以下のとおりです．

- ブロック転送命令，およびストリング・サーチ命令
- ビット操作命令(セット，リセット，テスト)
- インデックス・レジスタを使う命令

❸インデックス・レジスタがある

8080Aではインデックス操作をするときには，レジスタHLにレジスタBC，またはDEの内容をダブル・アッド(DAD)してアドレッシングをしていましたが，Z80にはインデックス・レジスタが二つあり，これを用いてインデックス・レジスタが示すアドレスの＋127～－128バイトの範囲をアクセスできるようになりました．

❹レジスタ・セットが2組ある

Z80ではアキュムレータ，フラグ・レジスタ，ならびにレジスタB～Lが2セットずつあり，命令によってどちらのレジスタ・セットを使用するかの指定ができます．したがって，一方をふつうのルーチンが使用し，他方を割り込みルーチンが使用するというようなことも可能であり，割り込みルーチンがレジスタをセーブする必要がなくなり，同ルーチンのオーバ・ヘッドを減らすことが可能です．

ただし，インデックス・レジスタ，およびスタック・ポインタは1セットしかありません．おそらく開発時のLSIの集積度の関係でしょうが，これらのレジスタも2セットあれば，Z80はさらに使いやすくなっていたことでしょう．

❺割り込みモードが三つある

Z80にはモード0～モード2という三つの割り込みモードがあります．モード0およびモード1は，8080Aと互換性のあるRSTタイプの割り込みモードです．Z80はリセットされるとモード0の割り込みモードになり，8080Aのソフトウェアとの互換性を保ちます．モード2はZ80オリジナルのベクトル式割り込みモードであり，128個の割り込みベクトルにより，メモリの任意のアドレスへ飛びこむことのできるモードです．

❻ 5V単一電源で動作する

8080Aは3種類の電源(＋5V，＋12V，－5V)を必要としましたが，Z80は＋5V電源だけで動作します．

❼クロックが簡単である

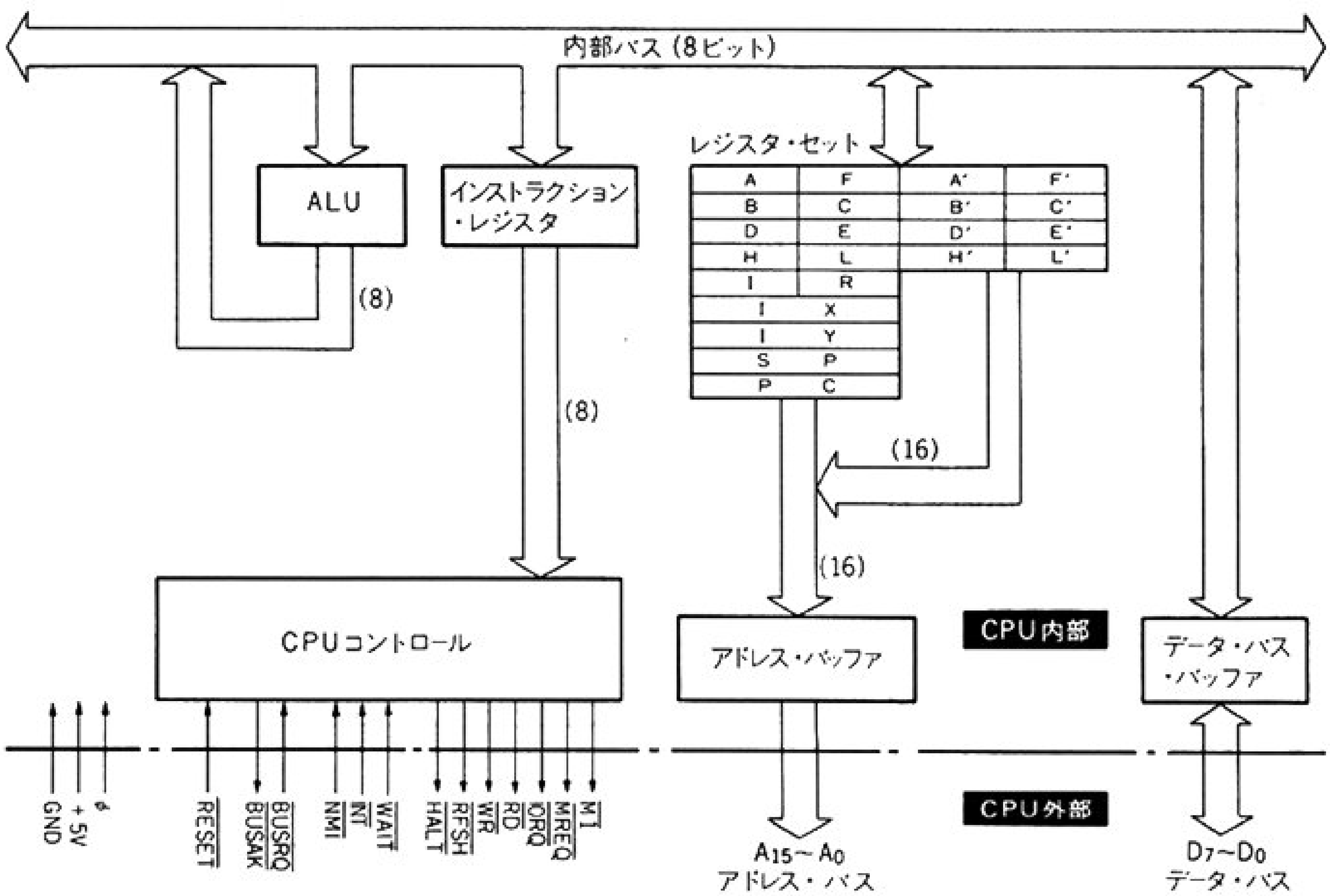
8080Aではϕ_1，ϕ_2という2相のTTLレベルではないクロックを必要としましたが，Z80はTTLレベルに近いシングル・フェーズのクロックだけで動作します．

Z80 CPUの構成

図1にZ80の内部構成を示します．以下に各構成要素の働きを説明します．

■ALU (Arithmetic Logical Unit)

これは日本語でいうと“算術論理演算機構”といういかめしい名前になりますが，通常はALU(エーエルユー)と呼びます．この部分はCPU内部で必要とする論理演算，および算術演算を実行します．これらの演算の種類を細かく分



〈図1〉
Z80 CPUの内部構成

けると，次のようになります．

・加算(Add)
・減算(Subtract)
・論理積(AND)
・論理和(OR)
・排他論理和(Exclusive OR)
・比較(Compare)
・左右シフト，および回転(Shift and Rotate)
・インクリメント(Increment)
・デクリメント(Decrement)
・ビット・セット(Set Bit)
・ビット・リセット(Reset Bit)
・ビット・テスト(Test Bit)

■インストラクション・レジスタ

インストラクション・レジスタ(Instruction Register)は，日本語では“命令レジスタ”と呼ぶこともあります．このレジスタにはCPUがメモリからフェッチ(Fetch：取り出す)した命令がストアされ，その内容にしたがって，CPUコントロール部が必要な制御信号を，CPU内部および外部に配ります．

■アドレス・バッファ(Address Buffer)

このアドレス・バッファはラッチ付きの16ビット3ステート・バッファです．CPUがメモリやI/Oをアクセスするときのアドレス，およびメモリ・リフレッシュ用のリフレッシュ・アドレスを出力する必要のあるときには，レジスタ・セットの中の適当なレジスタの内容を，このアドレス・バッファにセットし，アドレス・バス(A_0～A_{15})上に出力します．

■データ・バス・バッファ(Data Bus Buffer)

このデータ・バス・バッファは，CPUがメモリやI/Oと命令やデータのやりとりをするための双方向性3ステート・バッファです．CPUがメモリから命令やデータを読んだり，I/Oからデータを読んだりするとき，また外部から割り込みベクトルを読んだりするときには，このバッファはCPU外部→CPU内部の向きに信号を伝達し，CPUがメモリやI/Oに対してデータを書き込むときには，CPU内部→CPU外部の向きに信号を伝達します．

■CPUコントロール(CPU Control)

この部分はCPUの中枢部であり，他の構成要素はすべてこのCPUコントロールの制御下で動作します．レジスタ・セットの内容をアドレス・バッファやデータ・バス・バッファに送ること，読み込んできた命令をインストラクション・レジスタにストアし，その内容に応じてCPU内部の各構成要素を制御したり，CPU外部に対して必要な制御信号を出力したりするのも，このCPUコントロールの働きです．

またCPU外部から入力される信号，$\overline{\text{WAIT}}$(CPUを待たせる)，$\overline{\text{INT}}$/$\overline{\text{NMI}}$(割り込み要求)，$\overline{\text{BUSRQ}}$(バス・リクエスト)，それに$\overline{\text{RESET}}$(リセット)信号などを認識し，適当な動作を行うのもこのCPUコントロールの働きです．

■レジスタ・セット(Register Set)

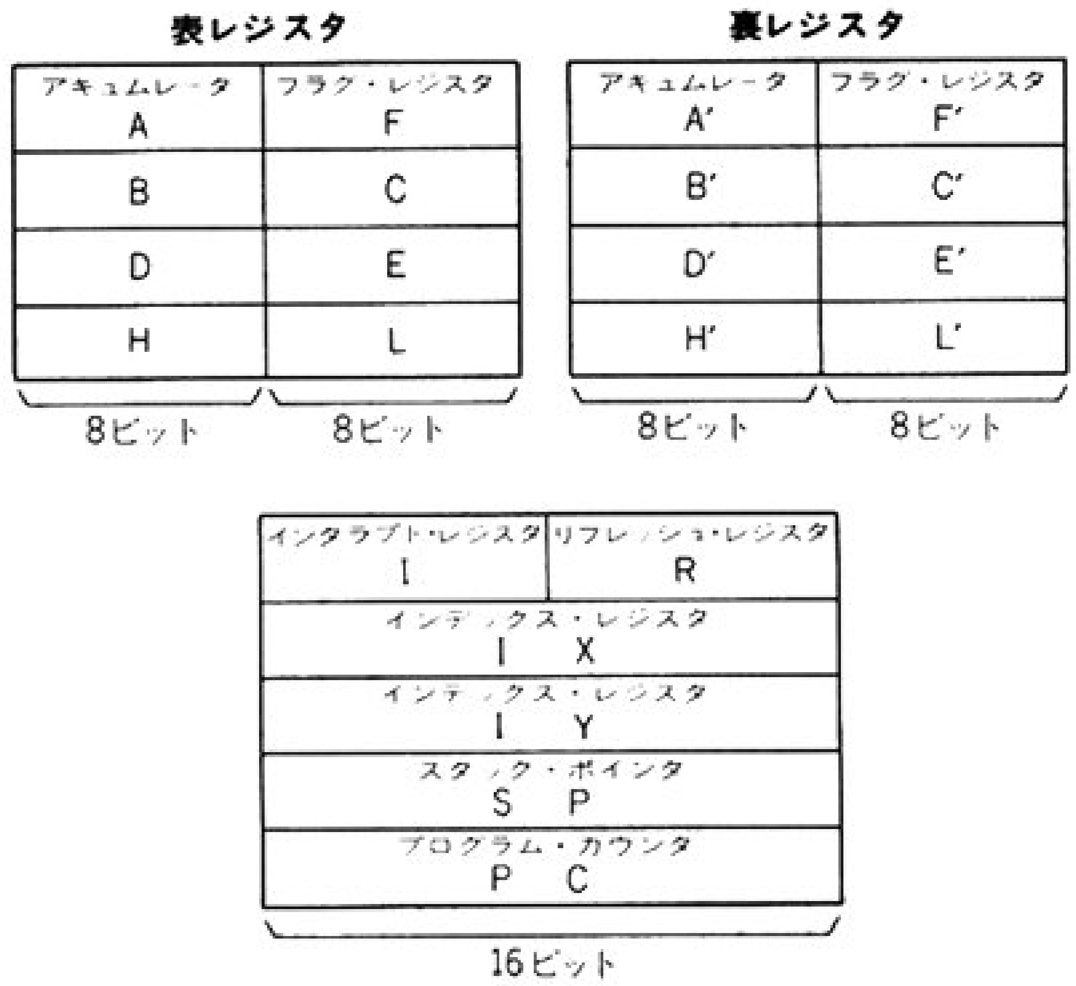
Z80のレジスタ・セットは，208ビット(8×26)のスタティックRAMにより構成されています．CPUコントロールの制御により，このレジスタ・セットの内容はアドレス・バッファに出力されたり，データ・バッファに出力されます(メモリやI/Oにレジスタの内容を書き込むとき)．

またメモリやI/Oから読み込まれたデータは，データ・バッファを介してレジスタ・セットに書き込まれます．さらにレジスタ・セットの内容に対して演算を施す必要のある場合には，レジスタ・セットの内容が内部バスを経由してALUに送られ，ALUの演算結果が内部バスを通じてレジスタ・セットに書き込まれます．

図2にZ80のレジスタ・セットのみを抜き出した図を示します．Z80にはアキュムレータ，フラグ・レジスタ，ならびにレジスタB～Lが2組あります．これら2組のレジスタ・セットに対して，通常私たちは“表レジスタ”，および“裏レジスタ”と呼んでいます．表レジスタには，A，F，B～Lの名が付けられており，裏レジスタには ′(ダッシュ)を付け，A′，F′，B′～L′と呼びます．

これら2組のレジスタ・セットの機能は完全に同等ですが，一時点にはどちらか1組のレジスタ・セットしか使えません．またアキュムレータ(A)，およびフラグ・レジスタ(F)とレジスタB～Lは，それぞれ独立に表または裏の選択が可能です．つまり，アキュムレータとフラグ・レジスタは表を使い，レジスタB～Lは裏を使う，というようなことが可能です．

〈図2〉Z80 CPUのレジスタ・セット

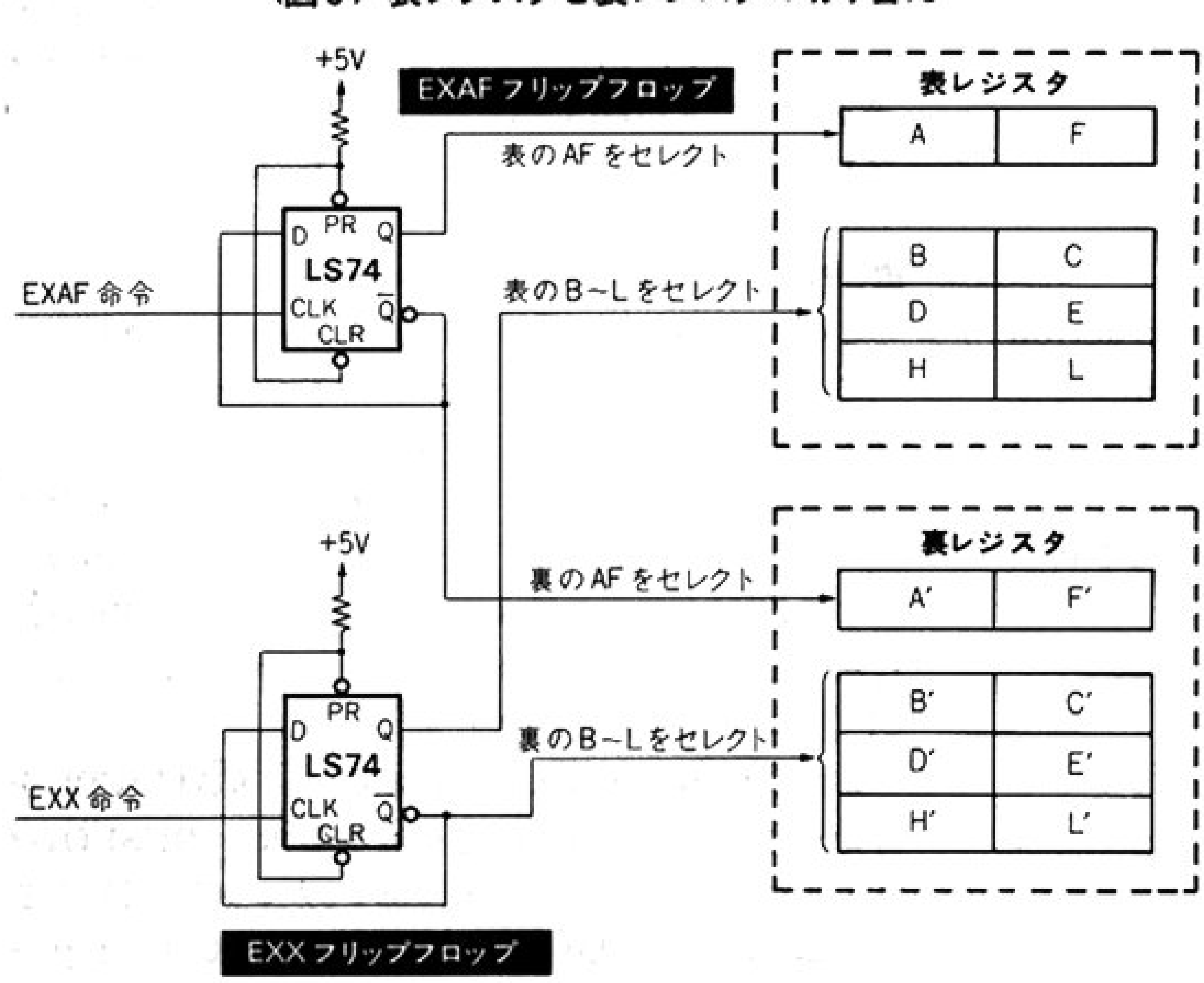


表レジスタと裏レジスタの選択は，EXAFという命令とEXXという命令によって行います．図3に表レジスタと裏レジスタの切り替えの概念図を示します．Z80の内部には二つのフリップフロップ，EXAFフリップフロップとEXXフリップフロップとがあり，前者はEXAF命令が実行されるごとにトグルし，後者はEXX命令が実行されるごとにトグルします．

Z80のパワーON時，またはリセット後のこれらのフリップフロップの状態がどうなっているかについて，プログラマは関知する必要はありません．場合によってはアキュムレータ，およびフラグ・レジスタは表側が選ばれ，レジスタB～Lは裏側が選ばれていたとしてもかまいません．Z80における表/裏レジスタの使用法は，どちらのセットを使っているのかというよりも，むしろ現在使っているセットを交換しさえすればよいのですから．

次に各レジスタの機能，および役割りについて，簡単に説明をします．

〈図3〉表レジスタと裏レジスタの切り替え



●アキュムレータ(A，A′)　算術演算(加算，減算)および論理演算(論理積，論理和，排他論理和)などのいわゆる“演算”は，すべてこのアキュムレータ内のデータに対して施され，演算結果はこのアキュムレータにストアされます．また，メモリやI/Oとのデータのやりとりにもこのレジスタが多く使われます．

●フラグ・レジスタ(F，F′)　算術演算，

論理演算，シフト/ローテイト演算，ビット・テストなどの演算結果(ゼロ/ノンゼロ，キャリーの有無など)がストアされます．このレジスタの内容を，条件付きジャンプ命令により判定することができます．

●レジスタB～L(B′～L′)　これら6個のレジスタは"汎用レジスタ"と呼ばれており，データの一時的なセーブ・エリヤ，カウンタなど，いろいろな目的で使われます．また，これらのレジスタは，二つずつペアにして(BとC，DとE，HとL)，16ビットのレジスタ・ペアとして使用することができ，メモリをアクセスする際のアドレス・ソースしても使うことができます．

またZ80の命令の中には，これらのレジスタに特殊な意味を持たせることがあります．たとえばメモリ同士のブロック・トランスファ命令(LDIR)では，レジスタ・ペアDEはデスティネーション・アドレス，HLはソース・アドレス，BCはレングス・カウンタとして用います．

このような例は他にも多くあります．図4にZ80の命令における，レジスタの特殊な使いかたをまとめます．

●インデックス・レジスタ(IX，IY)　Z80には二つの16ビット・インデックス・レジスタ，IXとIYがあります．これら二つのレジスタの機能はまったく対称です．IXまたはIYを用いてメモリをアクセスする命令には，ディスプレースメント部分があり，IXまたはIYの内容の＋127～－128番地の範囲のメモリを参照することができます．

●プログラム・カウンタ(PC)　プログラム・カウンタはパラレル・ロード可能な16ビット・カウンタであり，次に実行すべき命令のアドレスを保持しています．CPUはこのカウンタが示すメモリ・アドレスから命令を1バイト・フェッチするごとに，カウンタの内容を1だけ進めます．

ジャンプ命令やコール命令の実行時，および割り込みを受け付けたときには，新しい命令のアドレスがパラレル・ロードされます．このカウンタはリセット信号により，ゼロ・クリヤされますので，CPUはリセット後にはゼロ番地の命令から実行を開始します．

●スタック・ポインタ(SP)　スタック・ポインタの説明をする前に，まずスタックというものの概念を説明します．このスタックというのは，レジスタの内容を一時的にメモリにセーブしたり，サブルーチン・コールの際のリターン・アドレスをセーブしたり，割り込み時に割り込まれたプログラムのプログラム・カウンタの値をセーブしたりするのに用いるメモリ領域のことです．

スタック上にデータをセーブすることをプッシュ(PUSH)，スタックからデータをリストアすることをポップ(POP)するといいます．またプッシュ操作はメモリの高位番地から低位番地の向きへと行われますので，プッシュ・ダウン・スタックと呼ばれることもあります．

スタック・ポインタというのは，スタック上の次にプッシュすべきアドレスを保持している，16ビットのアップ/ダウン・カウンタであり，LD SP，nn他いくつかの命令により初期設定ができます．

以下図5により，スタック・ポインタとスタックの動作を説明します．

① まず初めの状態として，スタック・ポインタ(SP)は1000H*番地を指しており，スタック上には何もプッシュされていないものとします．

② レジスタ・ペアBCをスタック上にセーブする，PUSH BC命令を実行したとします．これによりZ80は，SPの内容を1だけ減じ，それが示すアドレス(図では0FFFH)にレジスタBをプッシュし，さらにSPの内容を1だけ減じたアドレスに，レジスタCをプッシュします．

したがって，この命令を実行した後のSPの値は，0FFEH(1000H－2H)となります．このようにZ80のSPは「次にプッシュすべきアドレス＋1」を示しています．

③ さらにレジスタ・ペアDEをスタック上にセーブする，PUSH DE 命令を実行すると，レジスタDおよ

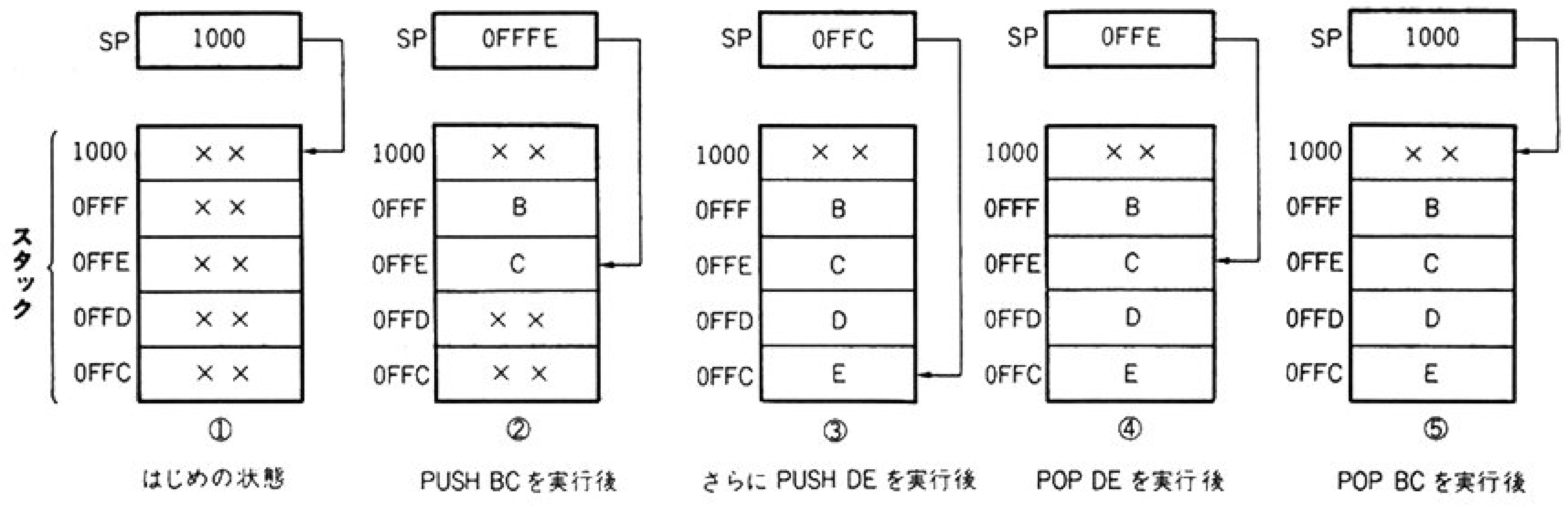
＊最後のHはHexadecimal(16進数)を意味する

〈図4〉汎用レジスタの特殊な使いかた

命　令	レジスタの使いかた
LDI LDIR LDD LDDR	HL：ソース・アドレス DE：デスティネーション・アドレス BC：レングス・カウンタ
CPI CPIR CPD CPDR	HL：ソース・アドレス BC：レングス・カウンタ
DJNZ	B：カウンタ

命　令	レジスタの使いかた
IN r，(C) OUT r，(C)	C：I/Oアドレス
INI INIR IND INDR OUTI OTIR OUTD OTDR	B：カウンタ C：I/Oアドレス HL：メモリ・アドレス

〈図5〉スタックおよびスタック・ポインタの説明図



びレジスタEがスタック上にセーブされ，SPの値はさらに2減じられて0FFCHとなります．

④ 次にスタック上からデータをレジスタE，およびDにリストアする命令，POP　DE　を実行すると，SPが示すアドレス(0FFCH)のデータがレジスタEにポップされ，その後SPの値は1だけ加えられます．そしてもう一度SPが示すアドレス(0FFDH)のデータがレジスタDにポップされたのち，SPの値に1が加えられます．

　その結果，SPの値は0FFFEHとなります．このようにポップの観点から見るとSPは，次にポップすべきメモリ・アドレスを指しています．

⑤ スタック上からデータをレジスタC，およびBにポップする命令，POP　BC　を実行すると，両レジスタにデータがポップされ，SPの値は初めの1000Hに戻ります．

以上のスタックおよびスタック・ポインタに関する説明は，PUSH命令およびPOP命令の実行に限りましたが，サブルーチン・コールの場合には，CALL命令実行時にリターン・アドレスがCPUにより自動的にプッシュされ，サブルーチンへジャンプし，サブルーチンの最後にRET命令を実行することにより，リターン・アドレスがPCにポップされ，CALL命令の次の命令に戻ることができます．

また割り込み発生時には，割り込まれたプログラムの次のアドレスが自動的にプッシュされ，割り込みルーチンの最後にRETI命令を実行することにより，割り込まれたプログラムの次のアドレスがPCにポップされますので，元のルーチンを続行することができます．

スタックという概念の持つすばらしさは，「ある一定の大きさを持ったメモリ領域(スタック)を何度でも再使用できる」という点にあります．プッシュすることにより，スタック領域の一部を借用しますが，ポップすることにより不要になった領域を他の使用のために返却するからです．スタック・ポインタはその貸し借りを管理する目付役といえましょう．

●インタラプト・レジスタ(I)　Z80がモード2の割り込みモードに設定されているときには，割り込みベクトルの下位8ビットは割り込み源から供給されます．

Z80は割り込みベクトルの上位8ビットとして，このインタラプト・レジスタの内容を用い，合計16ビットのメモリ・アドレスを構成し，割り込みルーチンのアドレス・テーブルから割り込みルーチンのアドレスを得，そこにジャンプします．このインタラプト・レジスタに値をロードするには，LD　I，A命令を用います．

●リフレッシュ・レジスタ(R)　Z80はダイナミックRAMをリフレッシュする機能を持っています．Z80はリフレッシュ・サイクルのときに，このレジスタの内容をアドレス・バスの下位8ビットに出力した後，下位7ビットの値を1だけ進めます．このレジスタのビット7の状態は変化しません．LD　R，A命令によりこのレジスタの値をセットし，LD　A，R命令によりこのレジスタの値を読むことは可能ですが，通常は，たとえダイナミックRAMを使う場合であっても，このレジスタに対して何らかのプログラミングをする必要があることはありません．

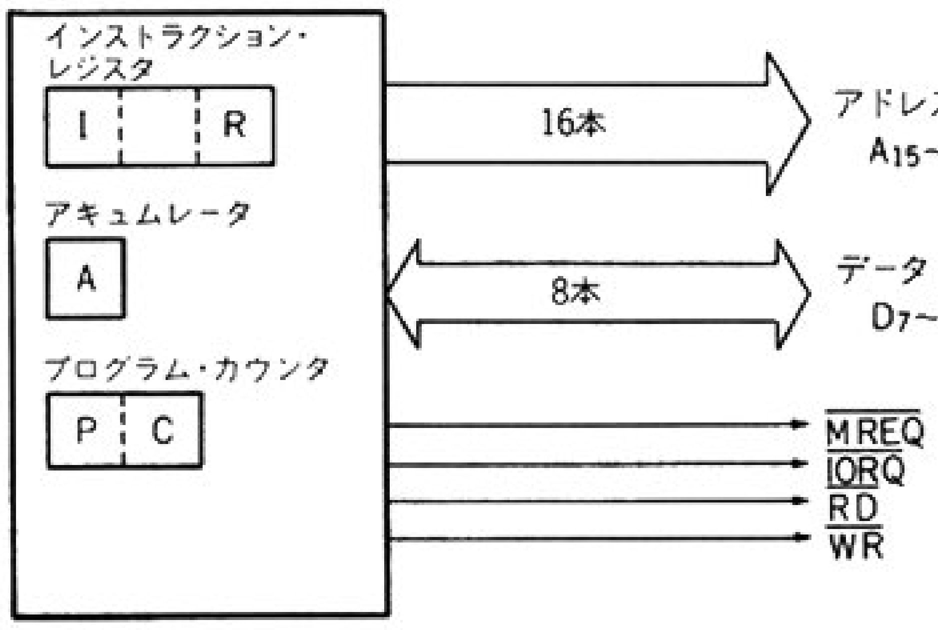
このレジスタの内容はZ80がリセットされると，ゼロ・クリヤされます．なぜそうなのか判じかねますが，Z80をリセットしても，ダイナミックRAMの内容を保持していたいような場合には，非常に**じゃま**な性質です．

Z80 CPUの基本動作

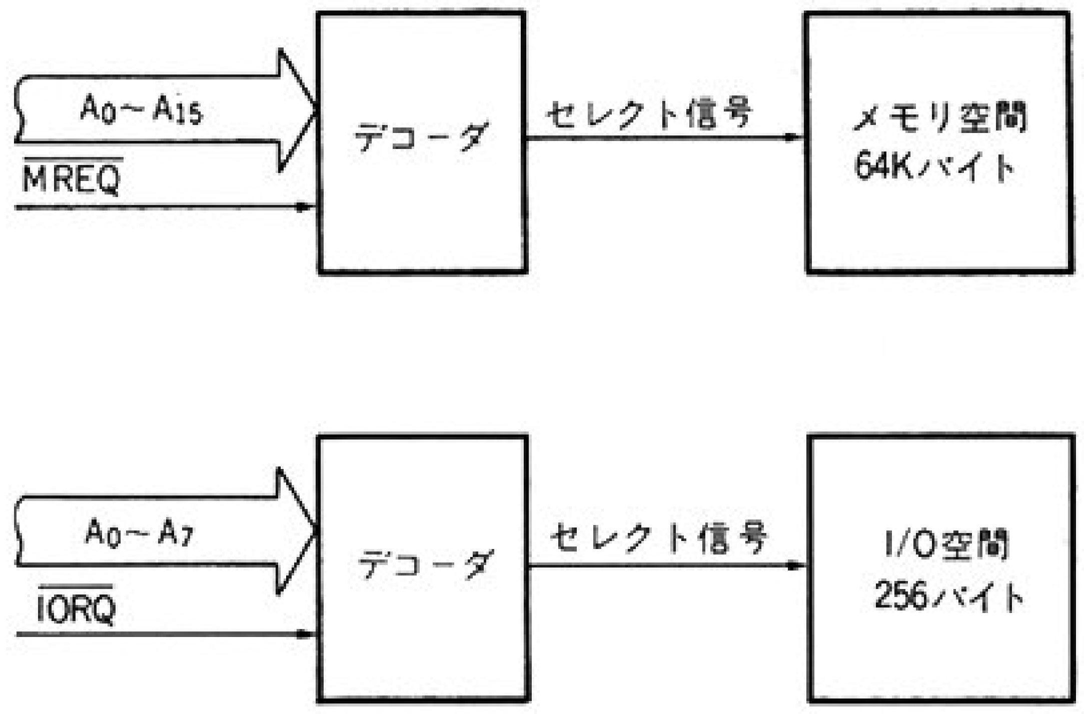
Z80 CPUに限らず，通常のCPUが行う基本動作は，

① メモリから命令をフェッチ(Fetch——取り出す)する．

〈図6〉Z80 CPUの簡略化モデル



〈図7〉Z80のメモリ空間とI/O空間



② フェッチした命令を解読し(Decode)，必要な機能を実行する．

であるということができます．

ここでは，Z80 CPUの基本動作を理解するために，Z80 CPUの簡略化モデルを作り，上記①，②の過程がどのように進んでいくかを見ていくことにします．

図6にここで用いる，Z80 CPUの簡略化モデルを示します．実際のZ80 CPUは40ピン構成であり，電源供給ピンとGNDを除くと，38本の信号源がありますが，このモデルではアドレス・バス16本，データ・バス8本，それにメモリおよびI/Oに関する制御線4本の合計28本だけです．

まずこれらの信号源の意味を簡単に説明しておきます．

●**アドレス・バス(A_{15}～A_0)**　CPUから出力される16本のアドレス・ラインであり，メモリから命令をフェッチするときや，メモリからデータを読む(Read)とき，ならびにメモリにデータを書き込む(Write)とき，このアドレス・バスを使用してメモリ・アドレスを指定します．

また，I/Oに対してデータを読み書きするときには，アドレス・バスの下位8本(A_7～A_0)上に，I/Oアドレスを出力します．

●**データ・バス(D_7～D_0)**　CPUがメモリから命令をフェッチするとき，メモリやI/Oからデータを読む場合，ならびにメモリやI/Oに対してデータを書き込むときに，このデータ・バスを用いてデータ(命令も含む)のやりとりをします．

このデータ・バスは双方向性であり，CPUからメモリまたはI/Oに対してデータを書き込むときには出力となり，メモリまたはI/Oからデータを読み込むときには入力となります．

●**$\overline{\mathrm{MREQ}}$(Memory Request)**　CPUがメモリやデータを読み込むとき，およびメモリに書き込むときに，この信号は“L”になります．

●**$\overline{\mathrm{IORQ}}$(I/O Request)**　CPUがI/Oとデータのやりとりをするときに，この信号は“L”になります．

●**$\overline{\mathrm{RD}}$(Read)**　CPUがメモリ，またはI/Oからデータを読むときに，この信号は“L”になります．

●**$\overline{\mathrm{WR}}$(Write)**　CPUがメモリ，またはI/Oに対してデータを書き込むときに，この信号は“L”になります．

以上の説明では“メモリ”および“I/O”という言葉(ことば)を何の説明もなしに使用してきましたが，ここで両者の意味を明確にしておきます．

Z80 CPUはデータ・バスを通じて外界とデータ(命令も含む)のやりとりをしますが，その外界というのはメモリ空間とI/O空間のことです．図7にZ80 CPUにおけるメモリ空間とI/O空間の概念図を示しました．

メモリ空間というのは，16本のアドレス・バス(A_{15}～A_0)と$\overline{\mathrm{MREQ}}$信号により選択される，64Kバイト(2^{16})の空間のことであり，これを通常“メモリ”と呼んでいます．

また，I/O空間というのは8本のアドレス・バス(A_7～A_0)と$\overline{\mathrm{IORQ}}$信号により選択される，256バイト(2^8)の空間のことであり，これを通常“I/O”と呼んでいます(使用するI/O命令を限定することにより――IN A,(n)やOUT (n),A命令を使用しない――I/O空間は64Kバイトになりますが，ここでは一応256バイトとしておきます)．

したがってメモリ空間とI/O空間の差異は，極端にいえば$\overline{\mathrm{MREQ}}$により選択されるか，$\overline{\mathrm{IOREQ}}$により選択されるかだけですので，I/O空間をまったく使用せず，メモリ空間の一部をI/O空間として使用することも可能です．こうした使いかたをメモリ・マップトI/O

(Memory Mapped I/O)と呼びます．

CPUによっては(たとえばモトローラの680X)，I/O空間という概念を持たず，初めからメモリ空間の一部をI/O空間として使用するように設計されたものもあります．このようなCPUの場合には，もちろんメモリ・マップトI/Oなどという用語も存在しません．

次にCPU内部に関しては，この簡略化モデルでは，プログラム・カウンタ(PC)とアキュムレータ(A)，それにインストラクション・レジスタ(IR)だけを設定します．

それでは前置きが長くなりましたが，Z80の基本動作の説明に入ります．この説明ではZ80が

①メモリからデータを読み取る命令を実行する場合

②メモリにデータを書き込む命令を実行する場合

③I/Oからデータを読み取る命令を実行する場合

④I/Oにデータを書き込む命令を実行する場合

を取りあげますが，いずれの場合も，命令は1000H番地にあるものとします．

①メモリからデータを読み取る命令

この種の命令の代表的な例は，LD　A，(nn)命令です．この命令は3バイトからなり，第1バイトはOPコード(命令の種類を示すコード)であり，第2，第3バイトはメモリ・アドレスです．例として次の命令を取りあげます．

LD　A，(4231H)　(3AH，31H，42H)

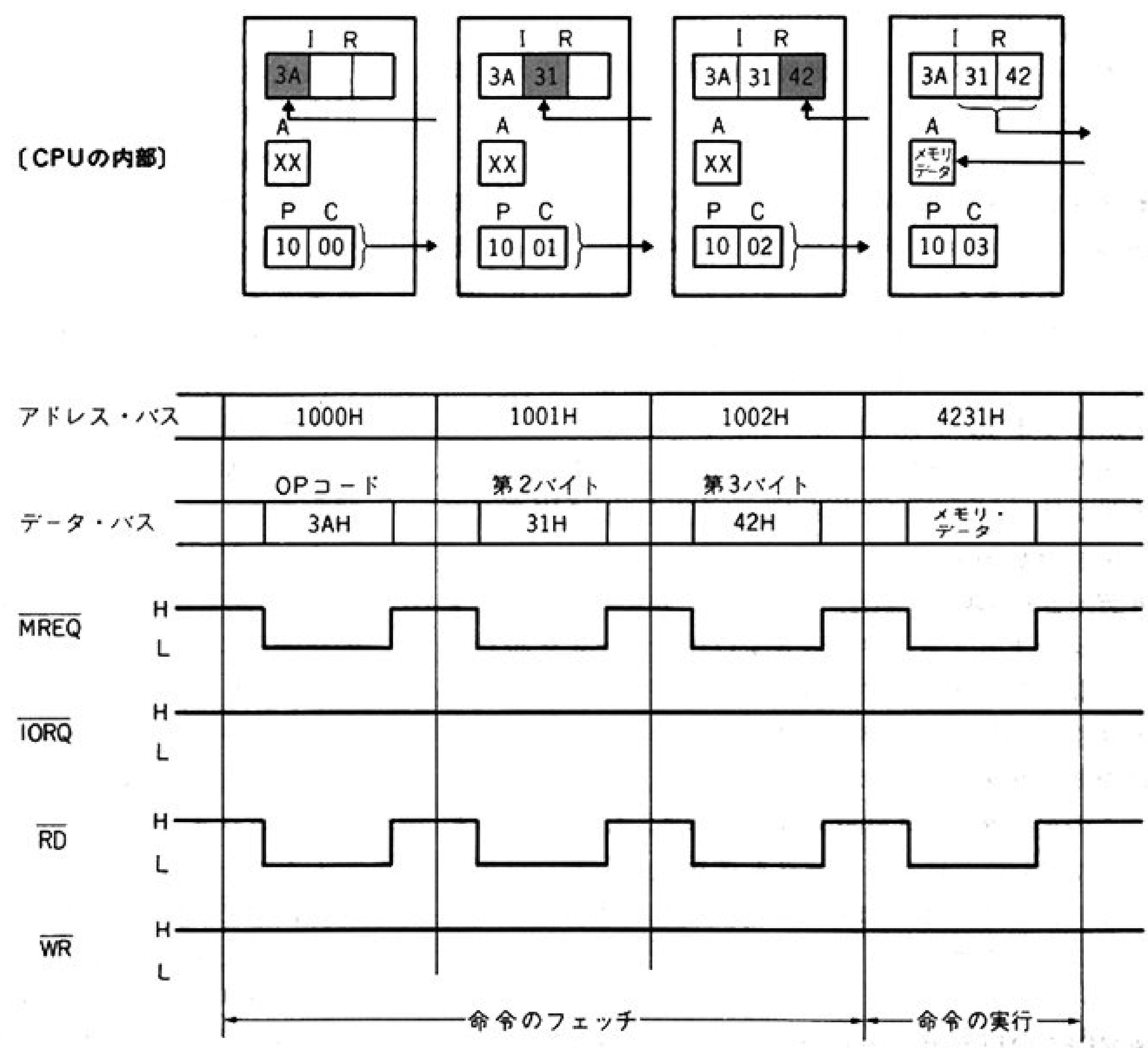
この命令のOPコードは3AHであり，メモリ・アドレスは4231H番地です．8080Aの場合と同様にZ80においても，メモリ・アドレスは下位バイト，上位バイトの順になります．この命令の機能はメモリの4231H番地のデータをアキュムレータに読み込むことです．

さきに命令は1000H番地にあると約束しましたので，OPコード3AHは1000H番地にあり，メモリ・アドレスの下位バイト31Hは1001H番地，上位バイト42Hは1002H番地にあることになります．

図8にこの命令を実行するときのCPUの動作を示します．CPUはまず命令をフェッチするためにPCの内容(1000H)をアドレス・バス上に出力し，メモリからデータを読むために，$\overline{MREQ}$と$\overline{RD}$を"L"にします．これにより，CPUは命令のOPコード(3AH)を読み取り，この命令は3バイトからなることを知ります．

次にCPUはこの命令の実行に必要なアドレス部分を得るために，1001H番地および1002H番地のデータを読み取ります．これでCPUはこの命令に関してすべきことはすべてわかりましたので，命令の実行に入ります．

〈図8〉
LD A，(4231 H)(3 A 3142)命令を実行するときのCPUの動作



CPUはOPコード3AHを見て，「この命令は命令の第2バイトおよび第3バイトで指定されたメモリ・アドレスのデータを読み取り，それをアキュムレータに入れる」ということを知っていますので，アドレス・バス上にメモリ・アドレス4231Hを出力し，$\overline{\text{MREQ}}$および$\overline{\text{RD}}$を“L”にし，4231H番地のデータを読み取り，それをアキュムレータにストアします．

②メモリにデータを書き込む場合

この種の命令の代表的な例は，LD (nn)，A命令です．この命令もやはり3バイトからなり，第1バイトはOPコード，第2バイトおよび第3バイトはメモリ・アドレスです．例として次の命令を取りあげます．

LD (1234H)，A (32H，34H，12H)

この命令のOPコードは32Hです．この命令の機能はアキュムレータの内容を，命令の第2バイトおよび第3バイトにより示されるアドレスのメモリに書き込むことです．第2バイトおよび第3バイトが反転しているのは，前出のLD A，(4231H)命令と同じです．したがってこの命令ではアキュムレータの内容をメモリの1234H番地に書き込みます．

図9にCPUがこの命令を実行するときの動作を示します．CPUはPCの内容1000Hをアドレス・バス上に出力し，$\overline{\text{MREQ}}$および$\overline{\text{RD}}$を“L”にして，命令のOPコード32Hを読み取ります．次にCPUは1001H番地および1002H番地のデータ(メモリ・アドレス)を読み取り，アキュムレータの内容を書き込むべきメモリのアドレス(1234H)を知ります．

そこでCPUはこのアドレス，1234Hをアドレス・バス上に乗せ，アキュムレータの内容をデータ・バス上に乗せた後に，$\overline{\text{MREQ}}$および$\overline{\text{WR}}$を“L”にして，データ・バス上のデータを指定されたメモリ(1234H番地)に書き込みます．

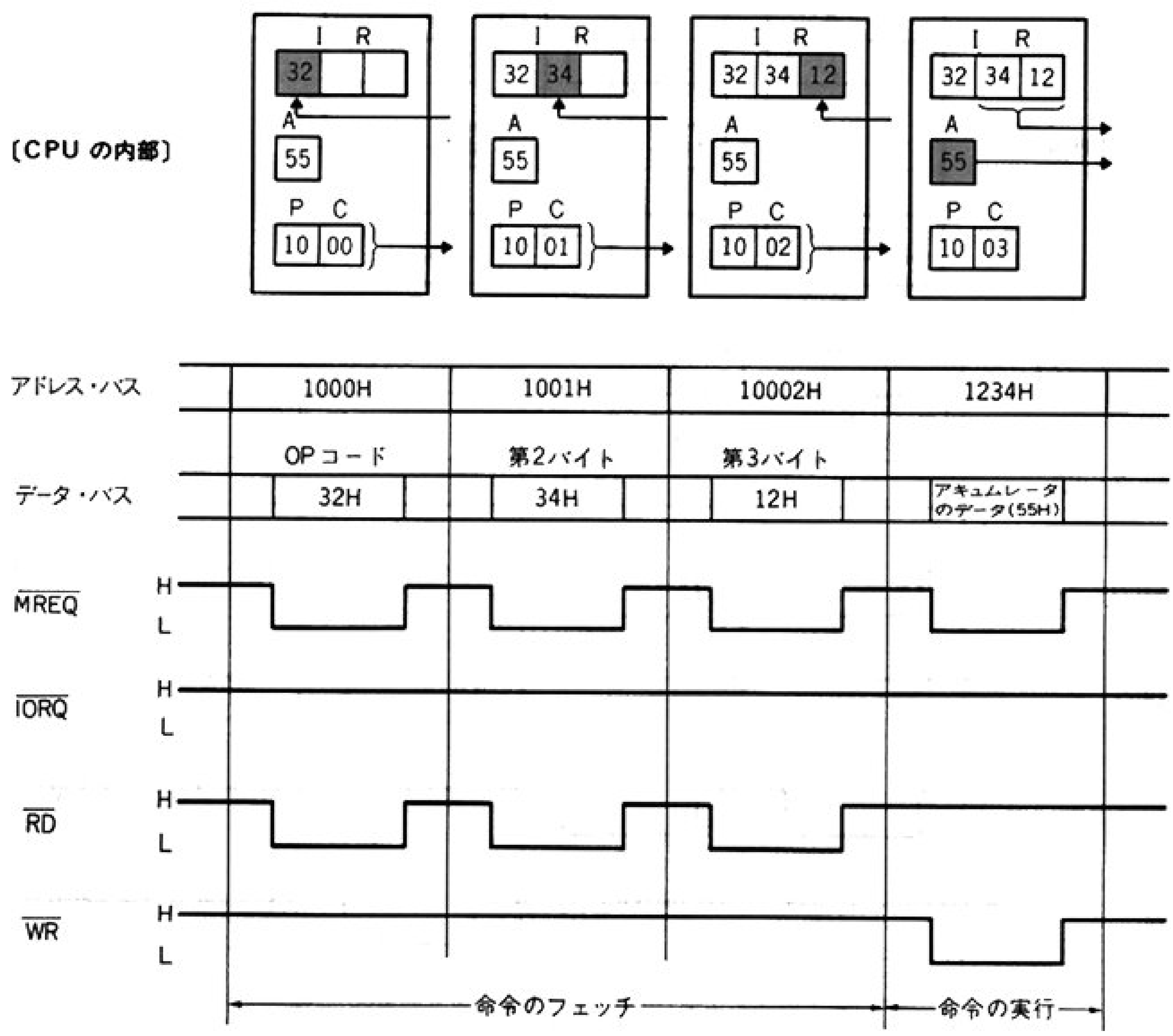
③I/Oからデータを読み取る場合

この種の命令の代表的なものは，IN A，(n)命令です．この命令は2バイトからなり，第1バイトはOPコード(DBH)であり，第2バイトはI/Oポート・アドレスです．例として次の命令を取りあげます．

IN A，(55H) (DBH，55H)

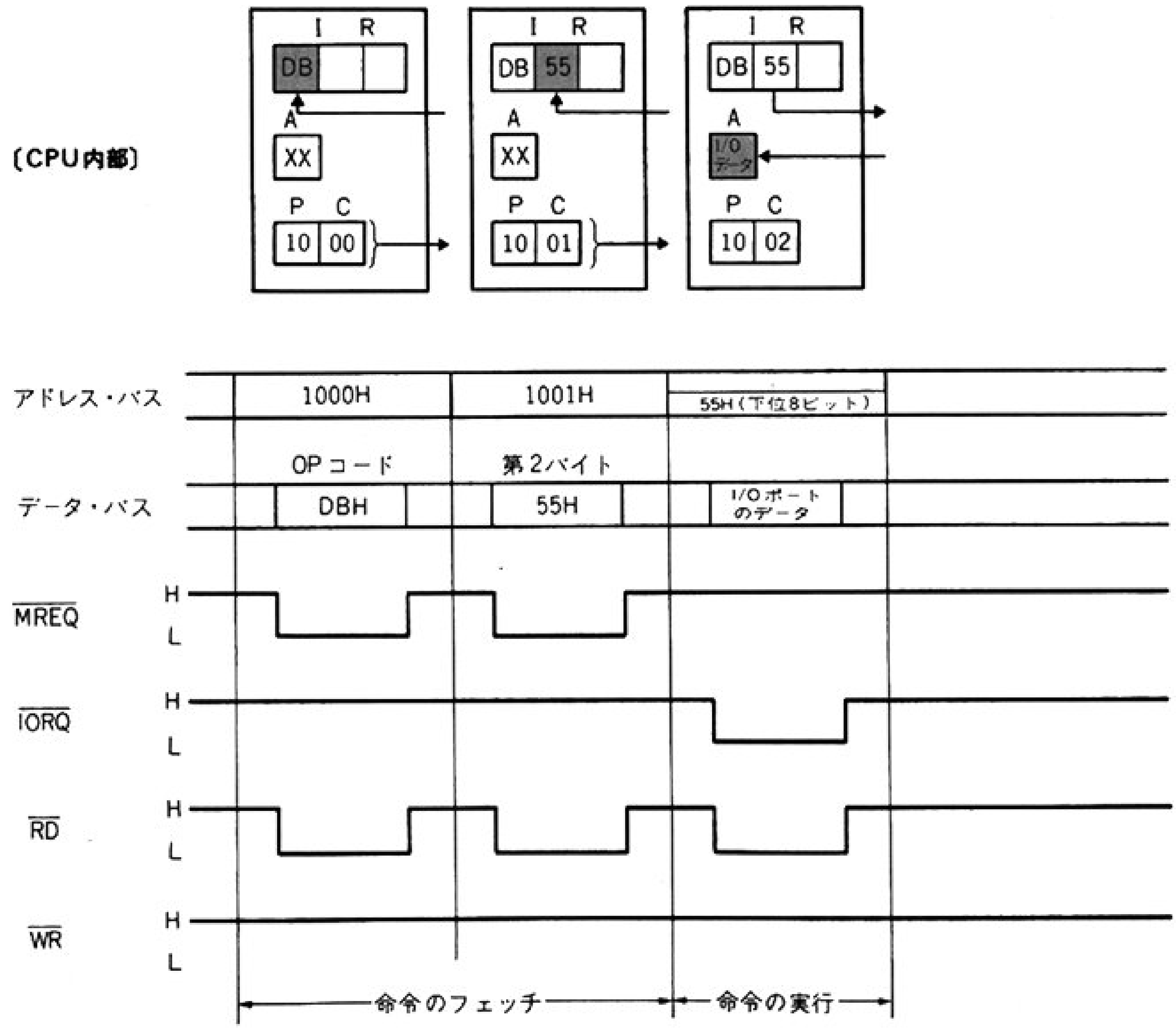
この命令のOPコードはDBHであり，I/Oポート・アドレスは55Hです．この命令の機能は，命令の第2バイトにより指定されたI/Oポートからデータを読み取り，それをアキュムレータにストアすることです．

図10にこの命令を実行するときのCPUの動作を示します．CPUはまずメモリの1000H番地，および1001

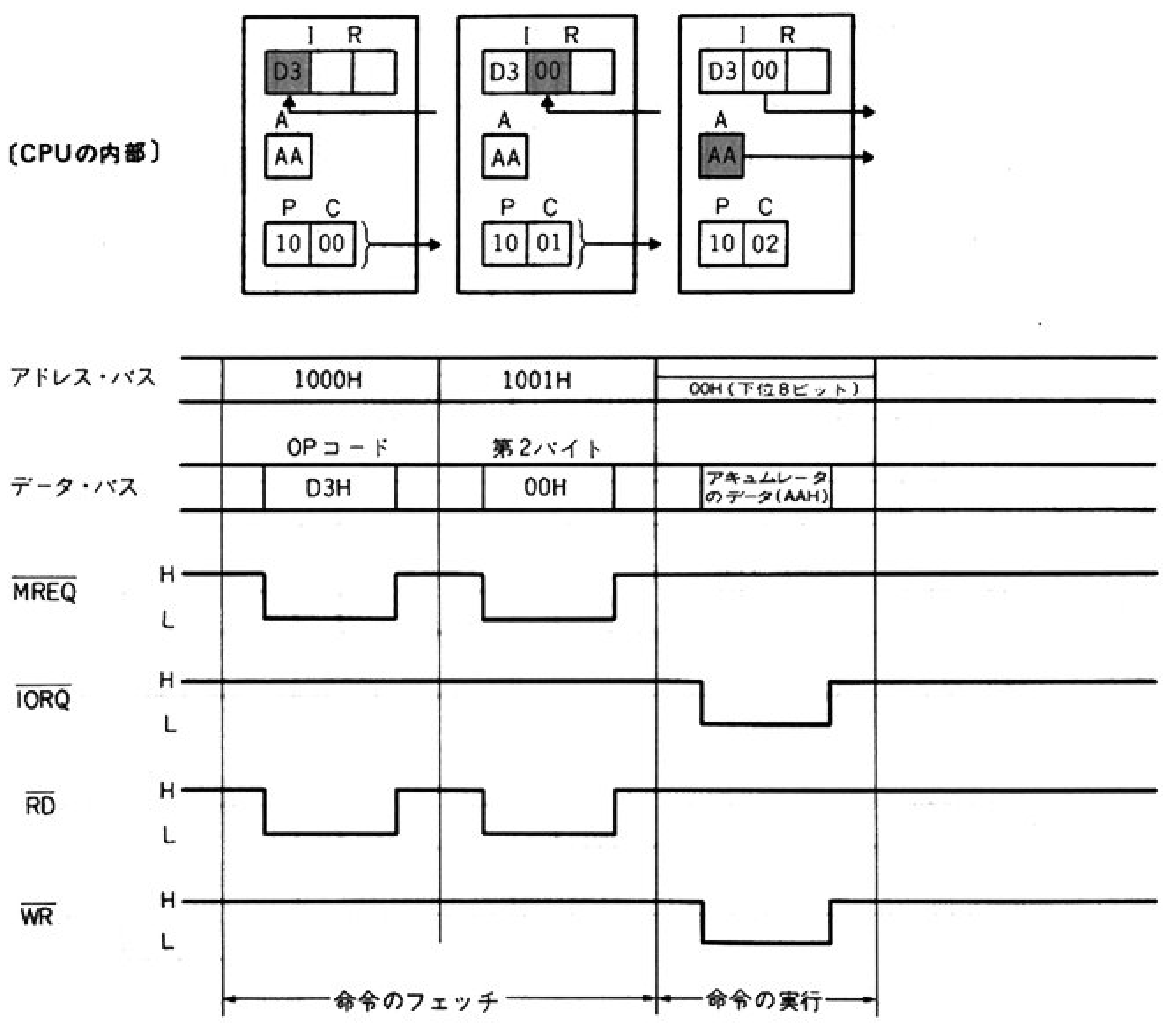


〈図9〉
LD(1234H)，A(323412)命令を実行するときのCPUの動作

〈図 10〉
IN A, (55 H) (DB 55)命令を実行するときのCPUの動作



〈図 11〉
OUT (00 H), A (D300)命令を実行するときのCPUの動作



H番地から命令を読み取ります．

次に命令の第2バイトと同じデータ(I/Oポート・アドレス)をアドレス・バスの下位8ビットに乗せ，$\overline{IORQ}$および$\overline{RD}$を“L”にしてポート55Hからデータを読み取り，それをアキュムレータにストアします．

④I/Oに対してデータを書き込む場合

この種の命令の代表的なものは，OUT　(n)，A命令です．この命令は2バイトからなり，第1バイトはOPコード(D3H)であり，第2バイトはI/Oポート・アドレスです．例として次の命令を取りあげます．

OUT　(00H)，A　(D3H，00H)

この命令はI/Oポート，00Hに対してアキュムレータの内容を書き込みます．図11にこの命令を実行する際のCPUの動作を示します．

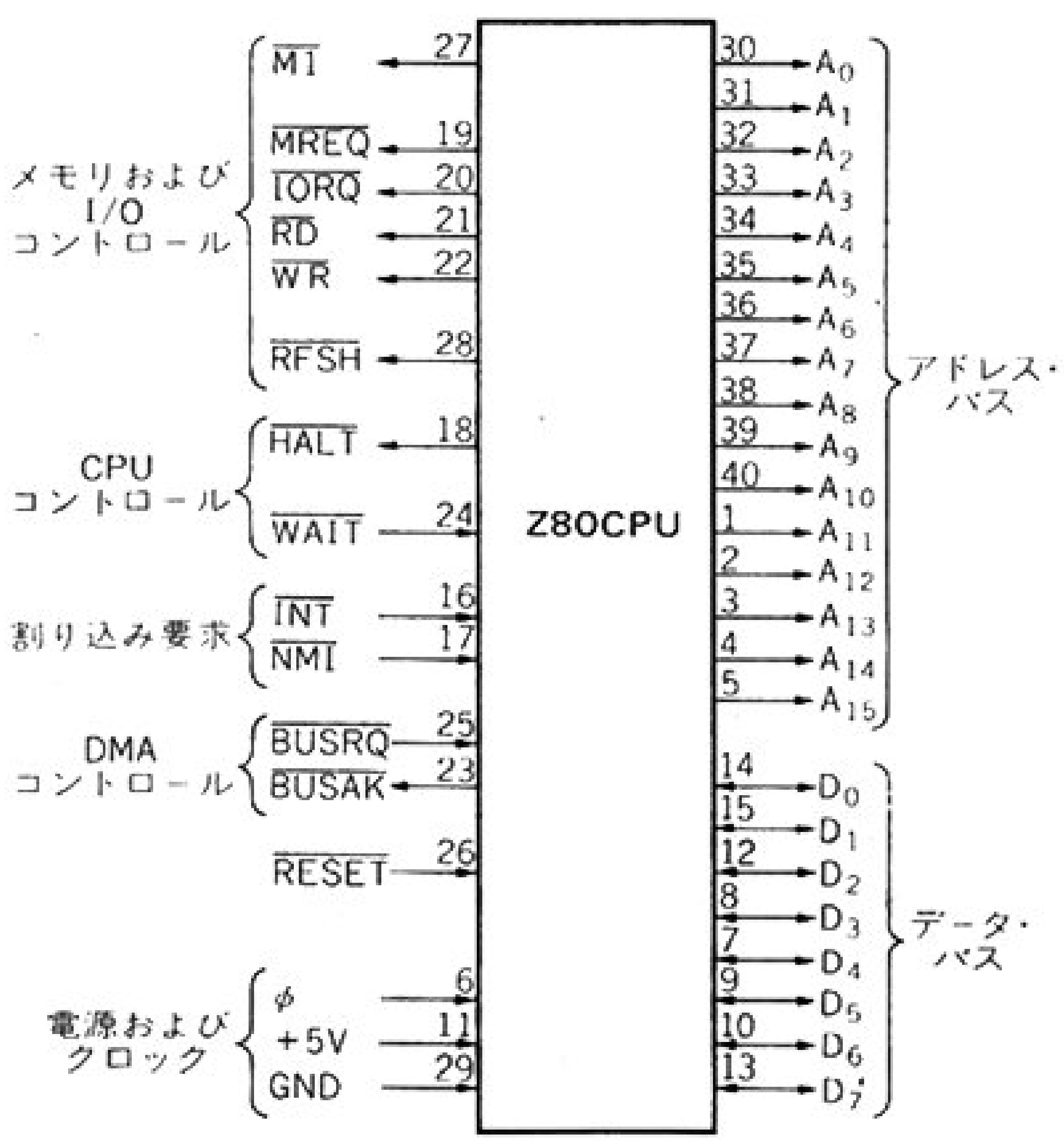
CPUはまず1000H番地および1001H番地のメモリから命令を読み取ります．そして命令の第2バイト(I/Oポート・アドレス)と同じデータをアドレス・バスの下位8ビットに乗せ，アキュムレータの内容をデータ・バス上に乗せた後に，$\overline{IORQ}$と$\overline{WR}$を“L”にして，アキュムレータの内容をI/Oポート00Hに書き込みます．

Z80 CPUの信号とタイミング

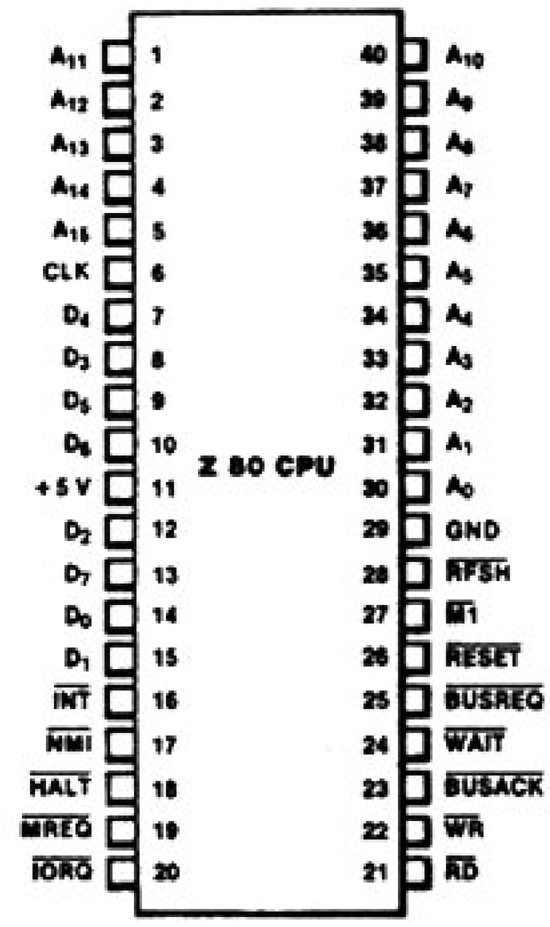
Z80 CPUの信号

Z80 CPUは40ピンDIPに納められており，40本の信号が出入りします．ここではZ80 CPUの各ピンに流れる信号について説明します．図12にZ80の各ピンの割り当てを示します．

〈図13〉 機能別に分けたZ80の信号



〈図12〉
Z80 CPUの各ピンの信号



この図はZ80のパッケージの各ピンの名称を，ピン①から㊵までピン番号順に示したものですが，図13にはZ80のピンに流れる信号を機能別に分けた図を示します．オシロスコープなどでZ80のピン状態を調べたりするときには図12が便利で，CPUを含むロジックを作ったりするときには，図13が便利です．図14にZ80の各ピンの機能を示します．

なおこれまでに“H”や“L”という用語を無造作に使ってきましたが，“H”というのは信号の電圧レベルが，2.0V以上のことをいい，“L”というのは0.8V以下の電圧レベルを意味します．

Z80 CPUのタイミング

Z80が命令を実行する場合，次に示す五つの過程の組み合わせにより行います．

① 命令のOPコードをメモリから読む過程
② メモリから命令のOPコード以下を読んだり，メモリからデータを読む過程
③ メモリにデータを書き込む過程
④ I/Oからデータを読む過程
⑤ I/Oにデータを書き込む過程

「Z80 CPUの基本動作」の項で取りあげた，LD A，(nn)命令の場合には，①②②②の過程を経て命令を実行し，LD(nn)，A命令の場合には①②②③の過程が行われました．またIN命令の場合には①②④，OUT命令の場合には①②⑤の過程が用いられていました．

これらの各過程のことを“マシン・サイクル”と呼びます．つまりZ80は，異なるマシン・サイクルの組み合わせにより命令を実行します．

Z80ではこれらの各マシン・サイクルに対して，次のような呼びかたをします．

① OPコード・フェッチ・サイクル(M1サイクル)

〈図 14〉 Z 80の信号説明　　　〔L：0.8V以下　H：2.0V以上〕

ピン名	信号名	入/出力	有効極性	機能
A_0〜A_{15}	Address Bus (アドレス・バス)	出力 3ステート	H	・16ビットのアドレス・バスであり，A_0がLSB，A_{15}がMSBである． ・CPUがメモリをアクセスするときには，このアドレス・バスにより，0000H〜FFFFH間の64Kバイトのメモリ空間中の1バイトをアドレッシングする． ・IN A，(n)およびOUT(n)，A命令の場合，I/Oアドレス(n)はA_0〜A_7上に出力され，00H〜FFH間の256バイトのI/O空間中の1バイトをアドレッシングする．なお，これらの命令の場合，A_8〜A_{15}上にはアキュムレータの内容が出力される． ・IN A，(n)およびOUT(n)，A以外のI/O命令では，レジスタCの内容がA_0〜A_7上に出力され，レジスタBの内容がA_8〜A_{15}上に出力されるので，64KバイトのI/O空間をアドレッシングできる． ・ダイナミックRAM用のリフレッシュ・アドレスはA_0〜A_6上に出力される． ・$\overline{\text{BUSAK}}$信号が"L"であるときには，このアドレス・バスは，ハイ・インピーダンスになる．
D_0〜D_7	Data Bus (データ・バス)	双方向性 3ステート	H	・8ビットの双方向性データ・バスであり，D_0がLSB，D_7がMSBである． ・CPUがメモリをアクセスするとき，およびI/Oをアクセスするときには，このデータ・バスを用いてデータのやりとりをする． ・$\overline{\text{BUSAK}}$信号が"L"のときには，このデータ・バスは，ハイ・インピーダンスになる． ・割り込みベクトルもこのデータ・バスを通じてCPUに送られる．
$\overline{\text{M1}}$	Machine Cycle One (M1サイクル)	出力	L	・この信号が"L"であると，CPUがM1サイクル(OPコード・フェッチ・サイクル)を実行していることを示す． ・割り込み認識シーケンスにおいては，この信号と$\overline{\text{IORQ}}$が同時に"L"となる．
$\overline{\text{MREQ}}$	Memory Request (メモリ・リクエスト)	出力 3ステート	L	・CPUがメモリ・アクセスをするとき，およびメモリ・リフレッシュ・サイクルに，この信号は"L"になる． ・$\overline{\text{BUSAK}}$信号が"L"のときには，この信号は，ハイ・インピーダンスになる．
$\overline{\text{IORQ}}$	I/O Request (I/Oリクエスト)	出力 3ステート	L	・CPUがI/Oアクセスをするときに，この信号は"L"になる． ・割り込み認識シーケンスにおいては，この信号とM1が同時に"L"になる． ・$\overline{\text{BUSAK}}$信号が"L"のときには，この信号はハイ・インピーダンスになる．
$\overline{\text{RD}}$	Read (リード)	出力 3ステート	L	・CPUがメモリ，またはI/Oからデータを読み取るときに，この信号は"L"になる． ・$\overline{\text{BUSAK}}$信号が"L"のときには，この信号はハイ・インピーダンスになる．
$\overline{\text{WR}}$	Write (ライト)	出力 3ステート	L	・CPUがメモリ，またはI/Oに対してデータを書き込むときに，この信号は"L"になる． ・$\overline{\text{BUSAK}}$信号が"L"のときには，この信号はハイ・インピーダンスになる．
$\overline{\text{RFSH}}$	Refresh (リフレッシュ)	出力	L	・CPUがダイナミックRAM用のリフレッシュ・アドレスをA_0〜A_6上に出力しているときに，この信号は"L"になる．外部回路はこの信号と，$\overline{\text{MREQ}}$が同時に"L"になったときに，リフレッシュ動作を行う．
$\overline{\text{HALT}}$	Halt (ホールト)	出力	L	・この信号はCPUがHALT命令を実行すると"L"になる．この後CPUは割り込みまたはNMIが発生するまで，先の命令を実行することはせず，NOP命令を繰り返す．
$\overline{\text{WAIT}}$	Wait (ウェイト)	入力	L	・CPUがメモリ，またはI/Oをアクセスしようとしたときの適当なタイミングに，この信号を"L"にするとCPUはウェイト・ステートに入り，この信号が"H"になるまでメモリ，またはI/Oのアクセスを先に延ばす．
$\overline{\text{INT}}$	Interrupt Request (割り込み要求)	入力	L	・この信号が"L"であると次の2条件 ① CPU内部の割り込みイネーブルFFがセットされている ② $\overline{\text{BUSRQ}}$信号が"H"である ときに，CPUはそのとき実行していた命令の終了時に割り込み要求を受け付け，$\overline{\text{IORQ}}$および$\overline{\text{M1}}$を"L"にして割り込みベクトルを外部から取り込む．

$\overline{NMI}$	Non Maskable Interrupt (ノン・マスカブル・インタラプト)	入 力	L	・この信号が"L"であると，CPU内部の割り込みイネーブルFFの状態にかかわらず，$\overline{BUSRQ}$信号が"L"でない限り，その時点に実行していた命令の終了時にNMIを発生する．この信号はネガティブ・エッジで有効である．
$\overline{RESET}$	Reset (リセット)	入 力	L	・この信号が"L"であると(最低3クロック期間)，CPUの状態はリセットされ次のようなことがらが行われる． ① プログラム・カウンタの値が0000Hになる ② 割り込みイネーブル・フリップフロップがリセットされる ③ インタラプト・レジスタ(I)が00Hになる ④ リフレッシュ・レジスタ(R)が00Hになる ⑤ 割り込みモードがモード0になる
$\overline{BUSRQ}$	Bus Request (バス・リクエスト)	入 力	L	・この信号が"L"であると，CPUはその時点に実行していたマシン・サイクルの終了時に，アドレス・バス，データ・バス，ならびに他の制御信号をハイ・インピーダンスにし，バスの使用を外部にゆだねる．
$\overline{BUSAK}$	Bus Acknowledge (バス・アクノウリッジ)	出 力	L	・CPUは$\overline{BUSRQ}$信号を認識し，バスをハイ・インピーダンスにした後に，この信号を"L"にしてバスが外部に対して"解放"されたことを示す．
ϕ	Phase Clock (フェーズ・クロック)	入 力	／	・CPUに対するシングル・フェーズのクロックであり，電気的条件の中のHレベルの最低値は$V_{CC}-0.6$Vであるので，通常のTTLによりドライブするときには，330Ω程度の抵抗でプル・アップする必要がある．

② メモリ・リード・サイクル
③ メモリ・ライト・サイクル
④ I/Oリード・サイクル
⑤ I/Oライト・サイクル

OPコード・フェッチ・サイクルというのは，命令の第1バイト(OPコード)をメモリから読み込むためのマシン・サイクルですので，どのような命令を実行する場合にも必ずあり，かつ一番最初のマシン・サイクルですので，M1(Machine Cycle 1)サイクルとも呼ばれます．

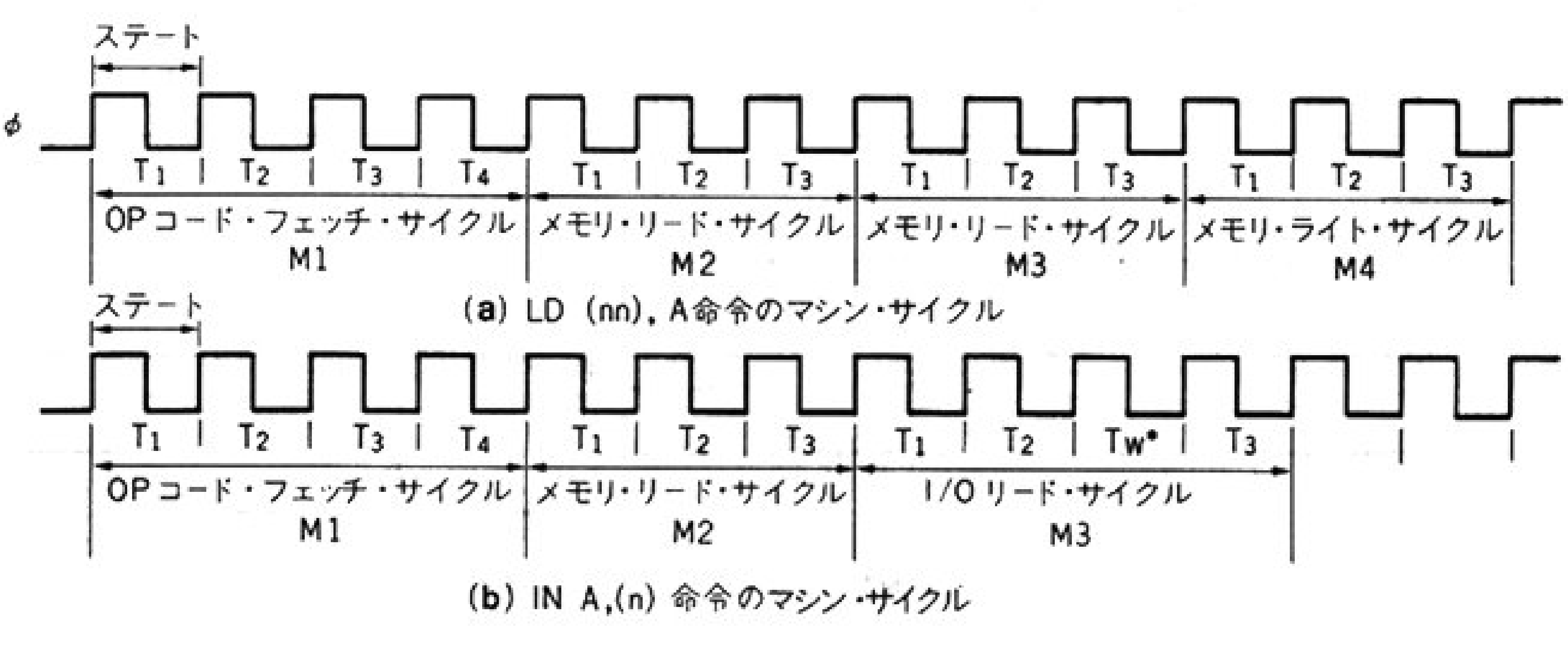
M1サイクルは各命令の実行時に必ず1回はありますが，常に1回であるとは限りません．Z80の命令の中には，DDH，EDH，またはFDHをOPコードとする命令がありますが，これらの命令のほとんどは第2OPコードを持っています．したがってこうした命令が実行されるときには，M1サイクルも2回あります．

図15にZ80のマシン・サイクルの例を示します．各マシン・サイクルはクロックの立ち上がり時点に開始され，M1サイクル，およびI/Oリード/ライト・サイクルは4クロック期間，メモリ・リード/ライト・サイクルは3クロック期間からなります．1マシン・サイクル中の各クロック期間に対して順に$T_1 \sim T_4$，T_W*ならびにT_Wという名を付け，1クロック期間のことをステートと呼びます．

以下にZ80のマシン・サイクルのタイミングについて詳細に説明します．以上に述べたマシン・サイクルのほかに，Z80には「割り込み認識サイクル」，「NMI認識サイクル」があり，これらについても説明をします．

OPコード・フェッチ・サイクル(M1サイクル)のタイミング

図16にOPコード・フェッチ・サイクルのタイミングを示します．このサイクルではT_1の始めにPCの内容



〈図15〉
Z80 CPUのマシン・サイクルの例

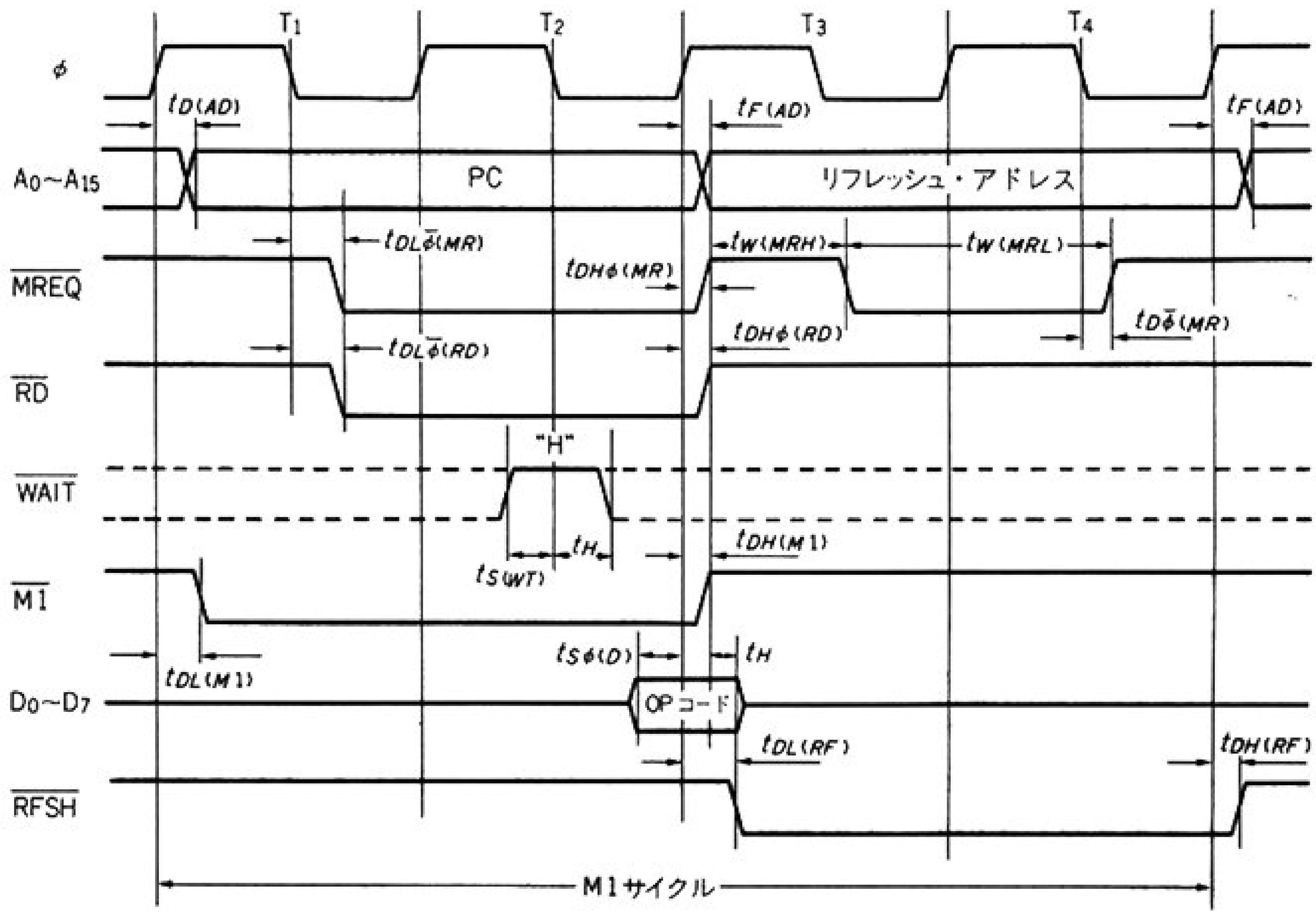
(次に実行すべき命令のアドレス)がアドレス・バス上に出力され，ほとんど同時に$\overline{M1}$信号が“L”になります．

次に同じT_1の中頃に，$\overline{MREQ}$および$\overline{RD}$信号が“L”になります．メモリから読み取られたデータ(OPコード)は，T_3のクロックの立ち上がり時点にCPU内に取り込まれます．

OPコードのフェッチ動作はT_1およびT_2の２クロック期間で終了しますが，その後CPUはT_3およびT_4を用いて，フェッチしたOPコードの解読を行いながら，ダイナミックRAM用のリフレッシュ動作に入ります．

これにはまずT_3の始めにアドレス・バスの下位７ビットにリフレッシュ・アドレスを乗せ，同時に$\overline{RFSH}$信号を“L”にします．次にT_3の中頃に$\overline{MREQ}$信号を“L”にして，ダイナミックRAMの周辺回路に対してリフレッシュ動作を促し，M１サイクルを終了します．

〈図 16〉
OPコード・フェッチ・サイクル(M１サイクル)のタイミング



〈図 17〉
1ウェイトが入ったM１サイクル

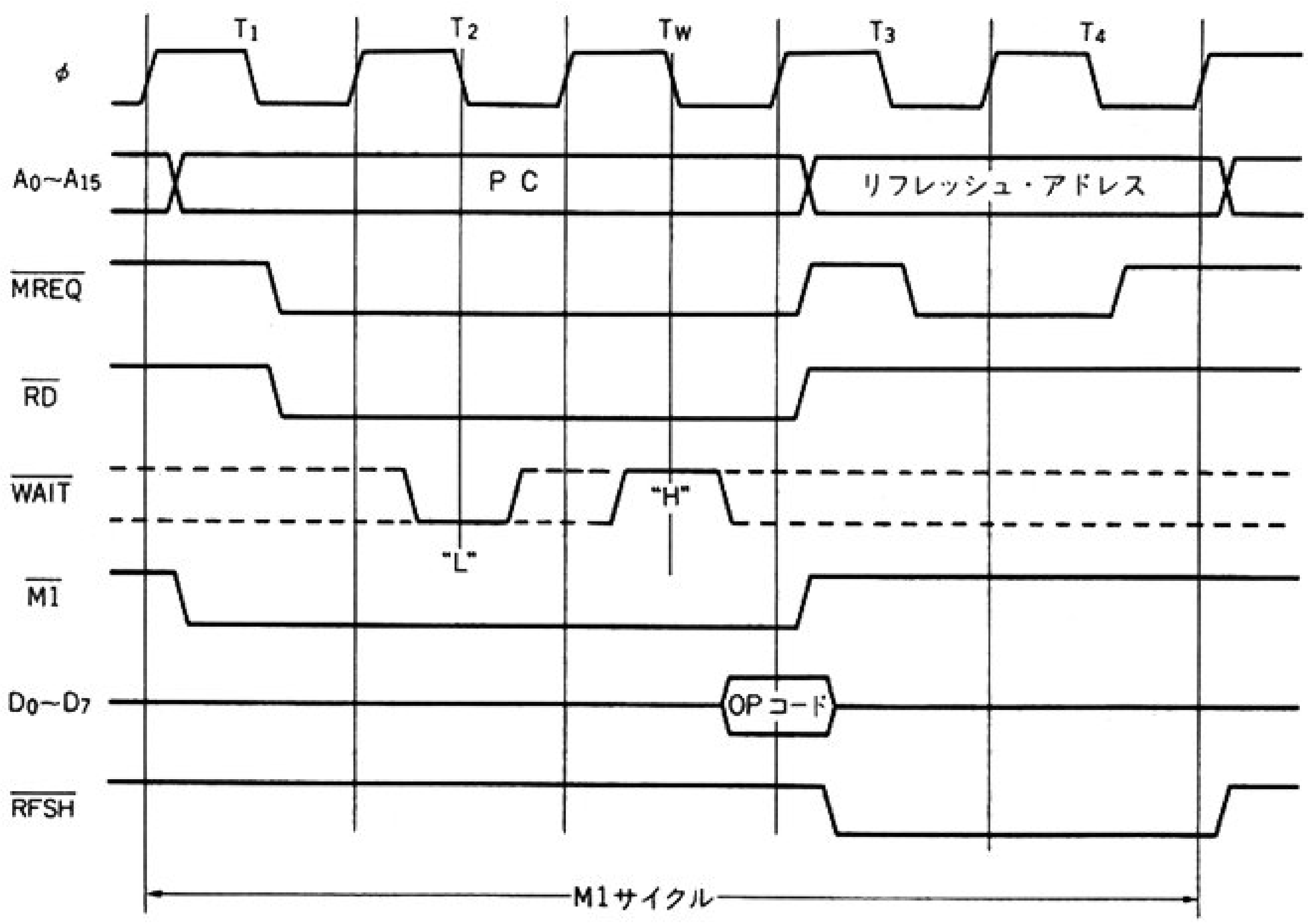


図16のタイミング・チャートにはクロックϕの立ち上がり，または立ち下がりと各信号の変化時点との間の遅延時間を表す記号が記入されています．たとえば$\overline{\mathrm{MREQ}}$の立ち下がりとT_1のクロックの立ち下がりとの間の遅延時間は，$t_{DL\overline{\phi}(MR)}$です．また$\overline{\mathrm{MREQ}}$と$T_3$の立ち上がりとの間の遅延時間は$t_{DH\phi(MR)}$です．

これらの遅延時間は，実際にZ 80 CPUを用いたシステムのロジック設計を行うときには，常に念頭におく必要があり，これらを無視して設計したシステムは，たとえ動いたとしても“まぐれ当たり”であり，通常は結構ひどい目に会うものです．遅延時間の実際の値については，後に示す「Z 80 CPUのAC特性」(図31，30ページ)をごらんください．

また図16ではT_2のクロックの立ち下がり時点の$\overline{\mathrm{WAIT}}$信号は“H”になっていますが，この時点に$\overline{\mathrm{WAIT}}$信号が“L”であると，CPUはT_2とT_3の間にウェイト・ステートT_Wを挿入します．そしてこのウェイト・ステート中のクロックの立ち下がり時点に再度$\overline{\mathrm{WAIT}}$信号の状態をサンプルし，その状態が“L”ですと次のクロック・サイクルも，ウェイト・ステートとなります．

このウェイト・ステートを抜け出してT_3ステートに進むためには，ウェイト・ステートのクロックの立ち下がり時点の$\overline{\mathrm{WAIT}}$信号の状態が“H”であることが必要です．

図17に$\overline{\mathrm{WAIT}}$信号により，一つのウェイト・ステートを挿入した場合のM1サイクルのタイミングを示し，図18に二つのウェイト・ステートを挿入した場合のタイミングを示します．

$\overline{\mathrm{WAIT}}$信号の状態は，T_2およびT_Wのクロックの立ち下がり時点にサンプリングされますが，実際にはセットアップ・タイム($t_{S(WT)}$)とホールド・タイム(T_H)があります．つまり$\overline{\mathrm{WAIT}}$信号はT_2またはT_Wのクロックの立ち下がり時点より，セットアップ・タイムだけ前に状態が確定していなければならず，また同じクロックの立ち下がり時点より，ホールド・タイムの期間，同じ状態を保持しておかなければなりません．

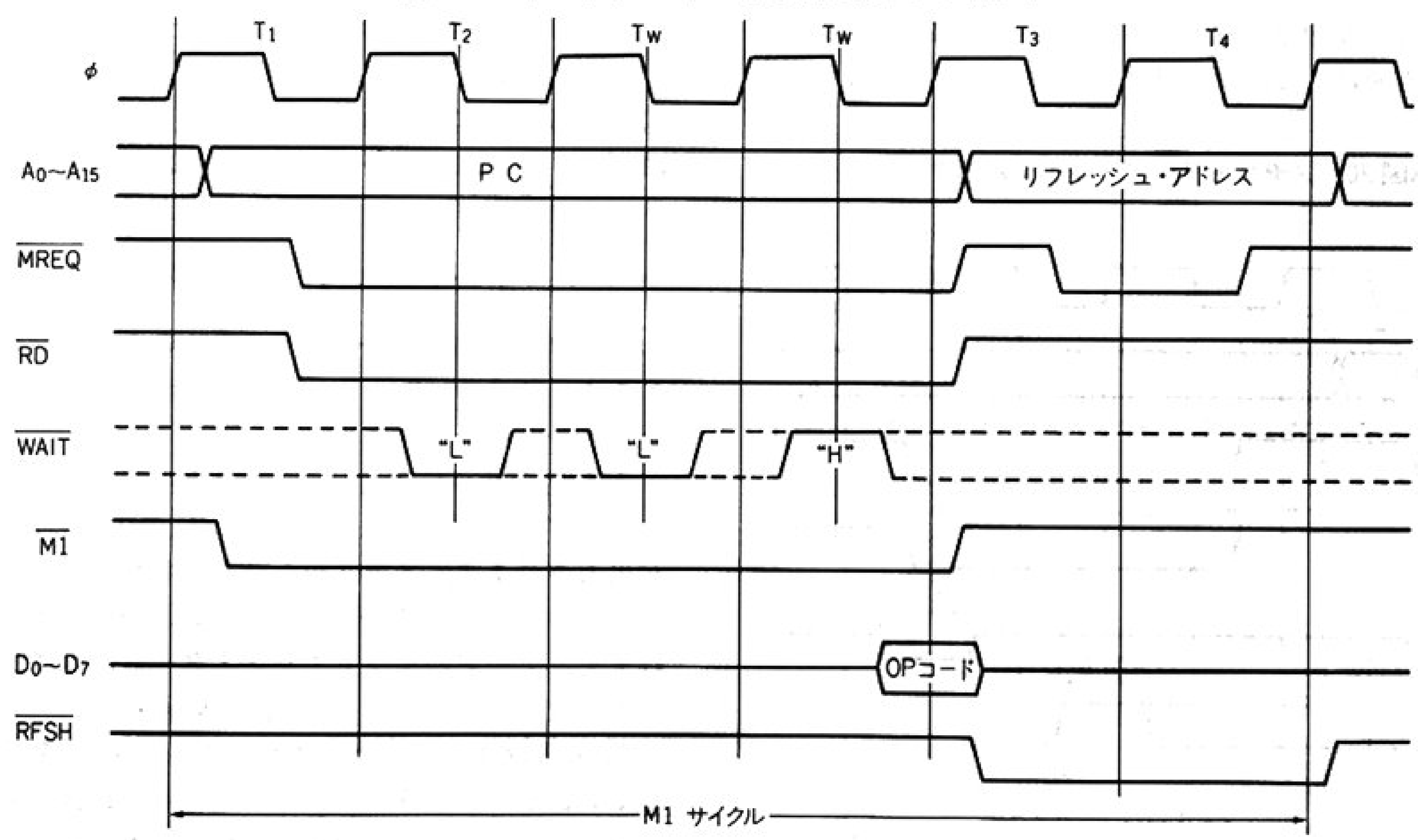
この約束を守らない場合には，$\overline{\mathrm{WAIT}}$信号を“L”にしてもCPUがウェイト・ステートに入らなかったり，ウェイト・ステート中に$\overline{\mathrm{WAIT}}$信号を“H”にしても，同ステートを抜け出せない，というような事態を招きかねません．

またT_2またはT_Wのクロックの立ち下がり時点を中心にした，セットアップ・タイムとホールド・タイムの期間以外のタイミングにおける$\overline{\mathrm{WAIT}}$信号の状態は，CPUの動作に影響を与えません．ここで示したタイミング・チャートでは，$\overline{\mathrm{WAIT}}$信号の状態がCPUの動作と無関係である期間を破線で示してあります．

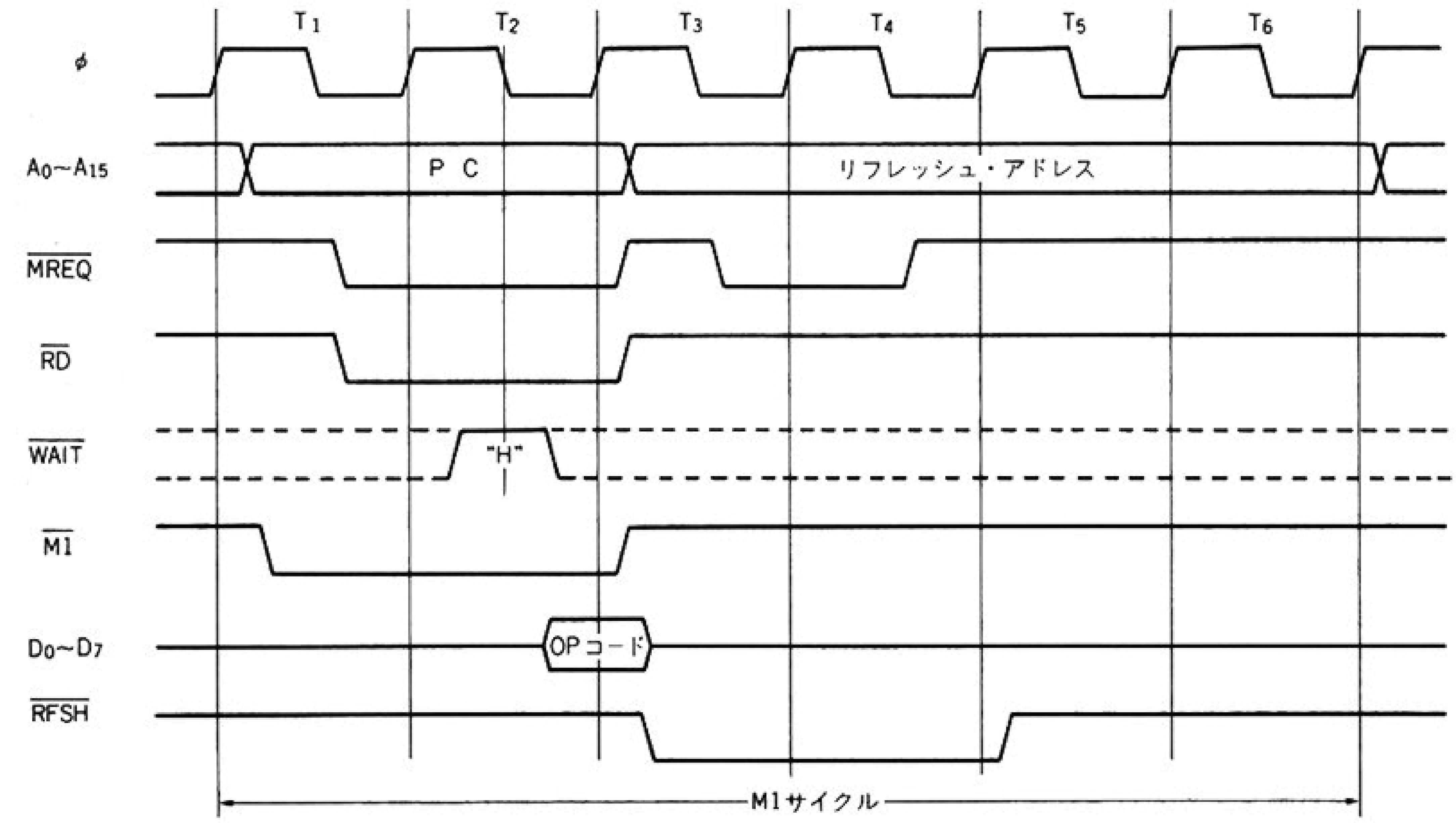
CPUはメモリから読み取ったデータ・バス上のデータ(OPコード)を，T_3のクロックの立ち上がり時点に，内部に取り込みますが，データ・バス上のデータに関しても，セットアップ・タイム($t_{S\phi(D)}$)とホールド・タイム(t_H)の約束があります．

つまり，メモリから読み取られたOPコードは，T_3の

〈図18〉 2ウェイト・ステートが入ったM1サイクル



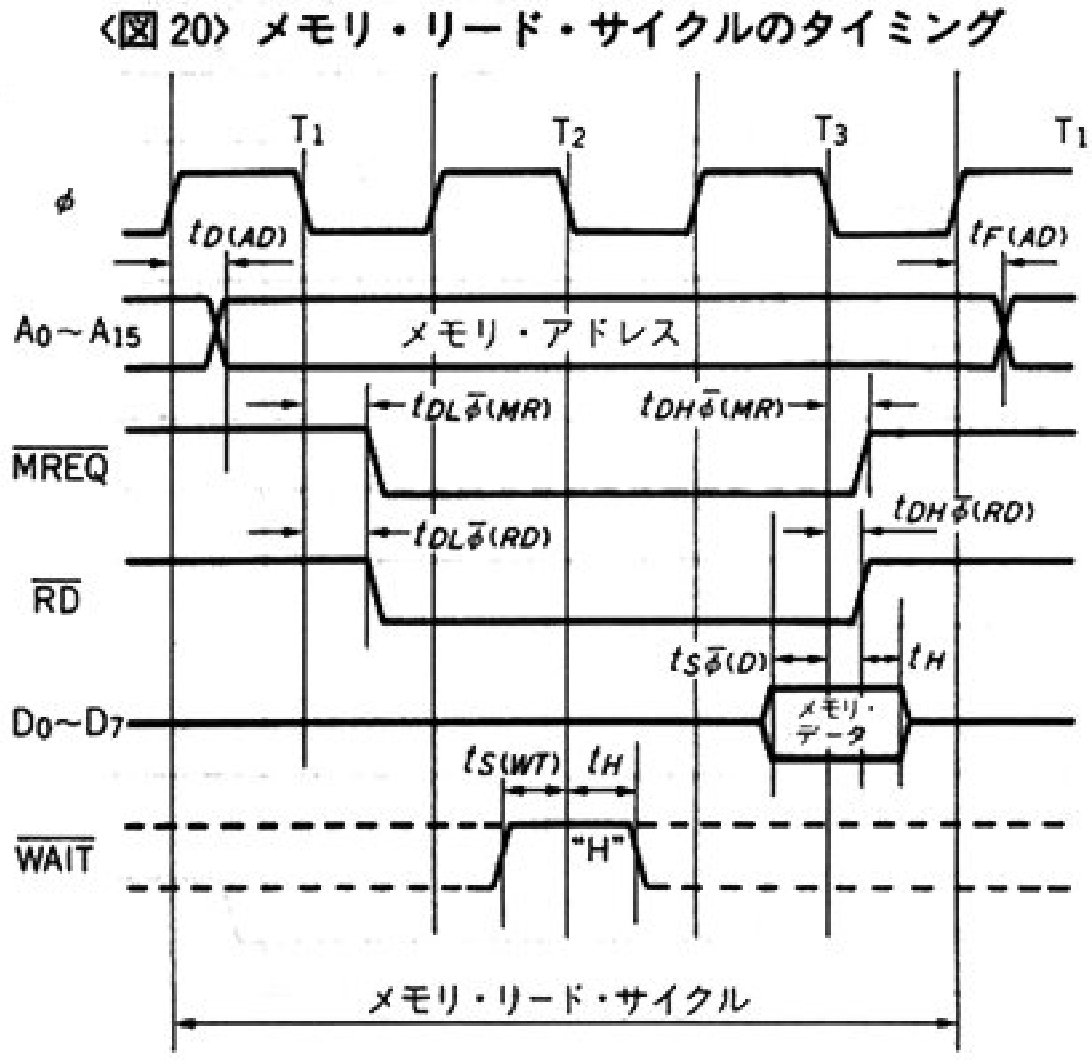
〈図19〉 6クロックからなるM1サイクル



クロックの立ち上がりより，セットアップ・タイムだけ前にデータ・バス上に乗っていなければならず，$\overline{RD}$信号が"H"になった後，ホールド・タイムの期間，データ・バスの状態を保持しておく必要があります．

今までに示したM1サイクルは，T_Wステートを含んだ場合でも，T_1～T_4で終わっていましたが，命令によってはT_1～T_5，あるいはT_1～T_6を用いる場合もあります．Z80の命令の中には，1バイトだけでかつM1サイクルだけで終わってしまうような命令があります．このような命令の場合，M1サイクルをT_5またはT_6まで引き延ばして，M1サイクル中に命令の実行を終了

〈図20〉 メモリ・リード・サイクルのタイミング



してしまいます．

たとえば条件付きRET命令（RET cc）において，条件が満足されない場合には，T_1～T_5にて命令の実行が終わります．またレジスタ・ペアに対する，INCまたはDEC命令もやはり1バイト命令ですが，この命令ではM1サイクルがT_1～T_6により構成され，M1サイクル中に命令の実行が終わります．T_5およびT_6を用いてレジスタ・ペアのインクリメント，またはデクリメント操作を行うからです．

図19に6クロックからなる，M1サイクルのタイミングを示します．図からわかるように，T_5およびT_6はCPU内部動作に使われるので，これらのステートでは$\overline{MREQ}$，$\overline{RD}$などのデータ・バスを制御する信号は出力されません．

メモリ・リード・サイクル のタイミング

図20にメモリ・リード・サイクルのタイミングを示します．メモリ・リード・サイクルは，T_1～T_3の3ステートからなります．

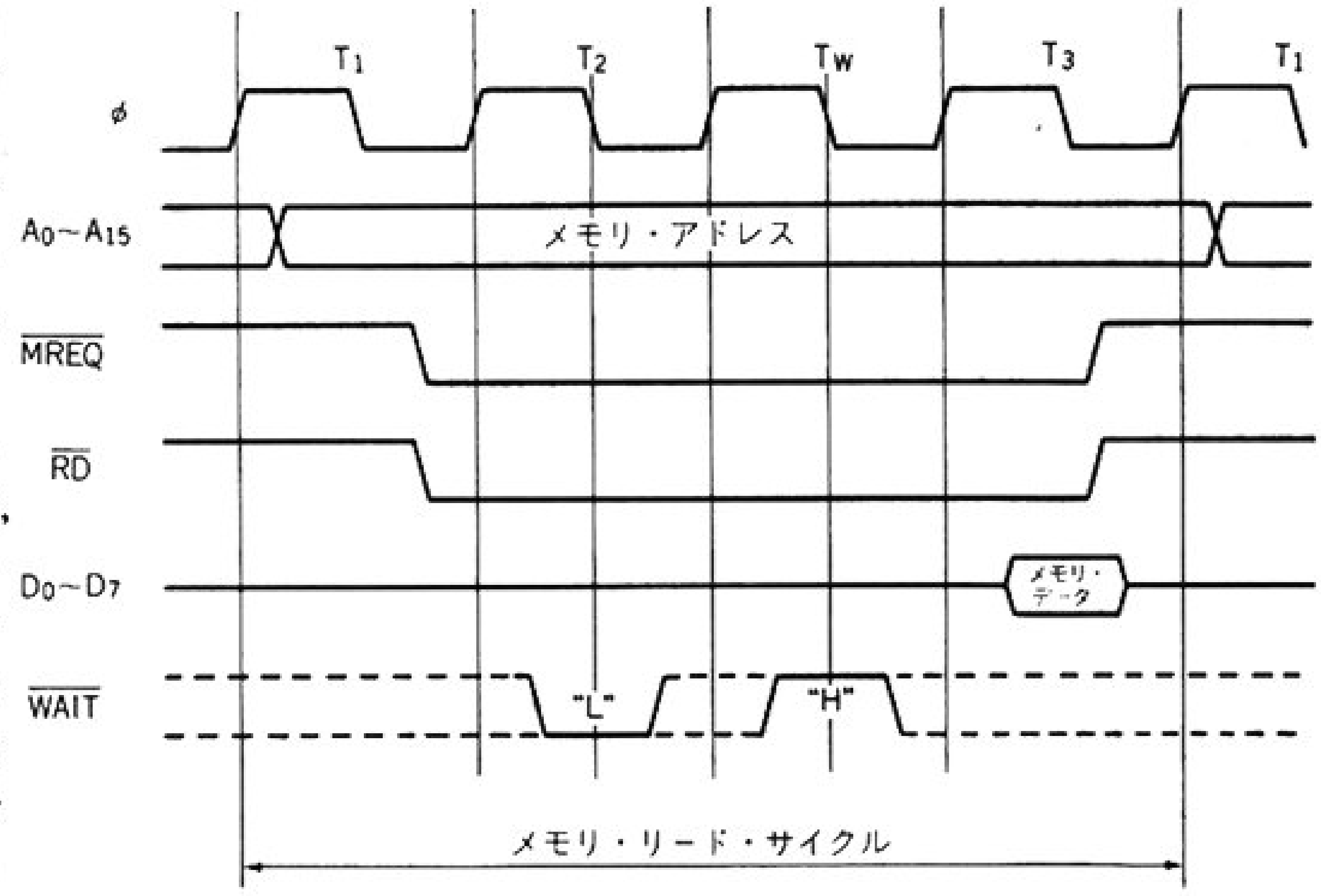
まずT_1の始めにアドレス・バス上に読み取るべきメモリのアドレスが出力され，次にT_1の中頃に$\overline{RD}$信号が出力されます．メモリから読み出されたデータは，T_3のクロックの立ち下がり時点に，CPU内に取り込まれます．データのセットアップ・タイム，およびホールド・タイムは，それぞれ$t_{S\phi(D)}$とt_Hです．

図20と図16（M1サイクルのタイミング）を比べてみておわかりのように，M1サイクルに比較してこのメモリ・リード・サイクルにおけるメモリ・アクセス

はずい分ゆるくなっています．両サイクルとも，アドレスおよび$\overline{\text{MREQ}}$信号が出力されるタイミングは同じですが，CPUがデータを内部に取り込むタイミングは約半クロック分異なり，M1サイクルではT_3の立ち上がり時点，メモリ・リード・サイクルではT_3の立ち下がり時点です．

したがって4MHzのクロックを用いた場合，クロック周期は250nsですので，両サイクルにおいて必要なアクセス・タイムは約125ns異なります．しかしながら通常のメモリは，命令のOPコードやOPコード以下の命令，データなどが混在しますので，M1サイクルのタイミングで動作できるように設計する必要があります．

〈図21〉1ウェイト・ステートを挿入したメモリ・リード・サイクル



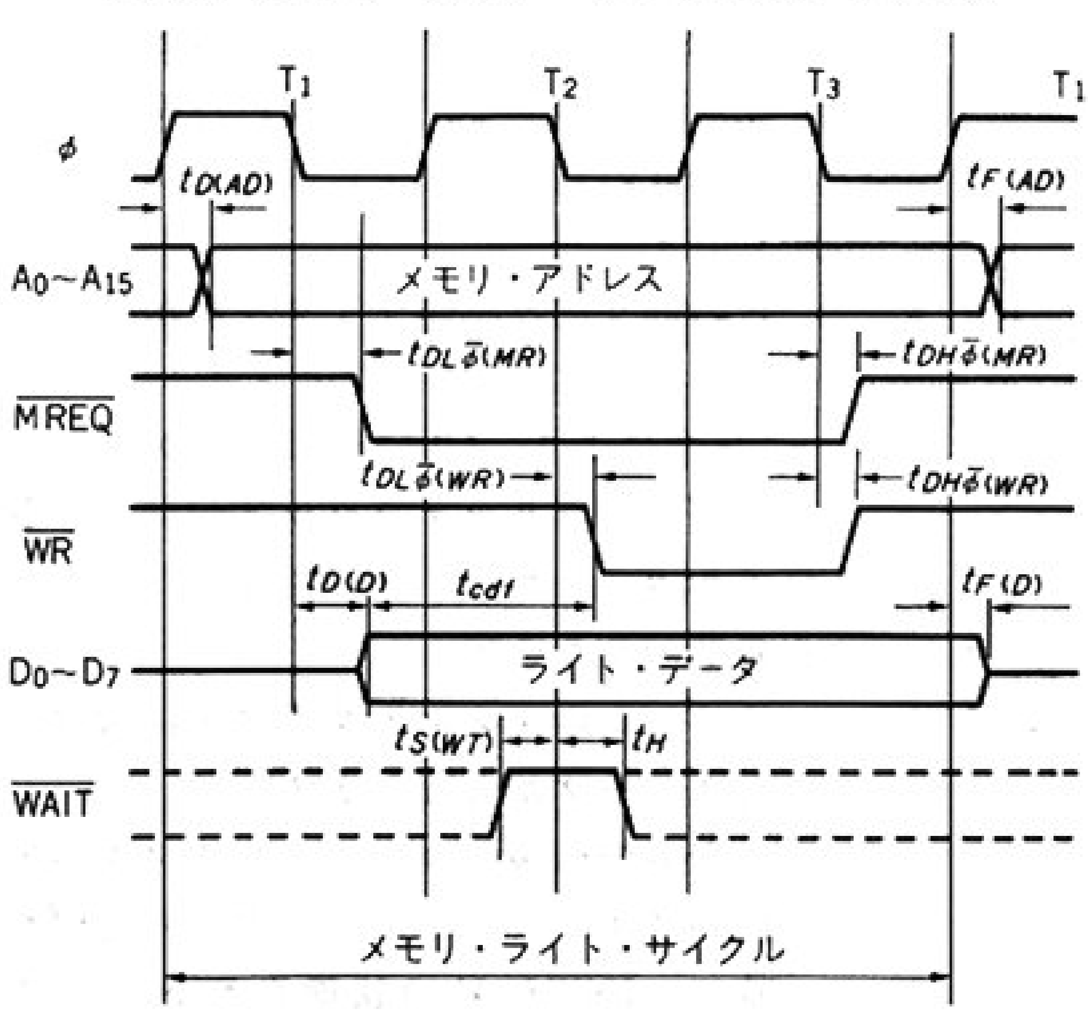
メモリ・リード・サイクルにおいても，T_2のクロックの立ち下がり時点に$\overline{\text{WAIT}}$信号の状態がサンプルされ，その状態が"L"であると，CPUはウェイト・ステートを挿入します．図21に1ウェイト・ステートを挿入した場合の，メモリ・リード・サイクルのタイミングを示します．

メモリ・ライト・サイクルのタイミング

図22にメモリ・ライト・サイクルのタイミングを示します．このサイクルもメモリ・リード・サイクルと同じく，T_1～T_3の3ステートからなります．まず，T_1の始めころに，データを書き込むべきメモリのアドレスが，アドレス・バス上に出力されます．

次に同じT_1の中頃に，データ・バス上に書き込むべきデータが出力され，同時に$\overline{\text{MREQ}}$信号が"L"になります．そして，次のT_2のクロックの立ち下がり時点に$\overline{\text{WR}}$信号が"L"になります．

〈図22〉メモリ・ライト・サイクルのタイミング



このメモリ・サイト・サイクルにおいても，T_2の立ち下がり時点に$\overline{\text{WAIT}}$信号の状態がサンプルされ，その状態が"L"であると，T_2とT_3の間にウェイト・ステートT_Wが挿入されます．図22ではT_2の立ち下がり時点の$\overline{\text{WAIT}}$信号の状態は"H"ですので，ウェイト・ステートはなく，T_2からT_3へとすぐに進んでいます．図23に1ウェイト・ステートを挿入した場合の，メモリ・ライト・サイクルのタイミングを示します．

I/Oリード/ライト・サイクルのタイミング

図24にI/Oリード・サイクルとI/Oライト・サイクルのタイミングを示します．両サイクルはT_1，T_2，T_W*，ならびにT_3の4ステートからなります．

I/Oリード・サイクルではT_1の始めにI/Oアドレスがアドレス・バス上に出力され，T_2の始め頃に$\overline{\text{IORQ}}$信号と$\overline{\text{RD}}$信号が"L"になります．そしてT_3の立ち下がり時点にデータ・バス上のデータ(I/Oポートのデータ)がCPU内に取り込まれます．

I/Oライト・サイクルでは，やはりT_1の始め頃にI/Oアドレスがアドレス・バス上に出力され，T_1の中頃にデータ・バス上にI/Oにライトすべきデータが出力されます．次にT_2の始めに$\overline{\text{IORQ}}$信号，および$\overline{\text{WR}}$信号が出力されます．

これらのマシン・サイクルでは，T_2の立ち下がり時点における$\overline{\text{WAIT}}$信号の状態とは関係なく，T_2とT_3の間に強制的にウェイト・ステートが挿入されます．CPUにより強制的に挿入されたウェイト・ステートのことをT_W*と呼び，$\overline{\text{WAIT}}$信号により挿入されたウェ

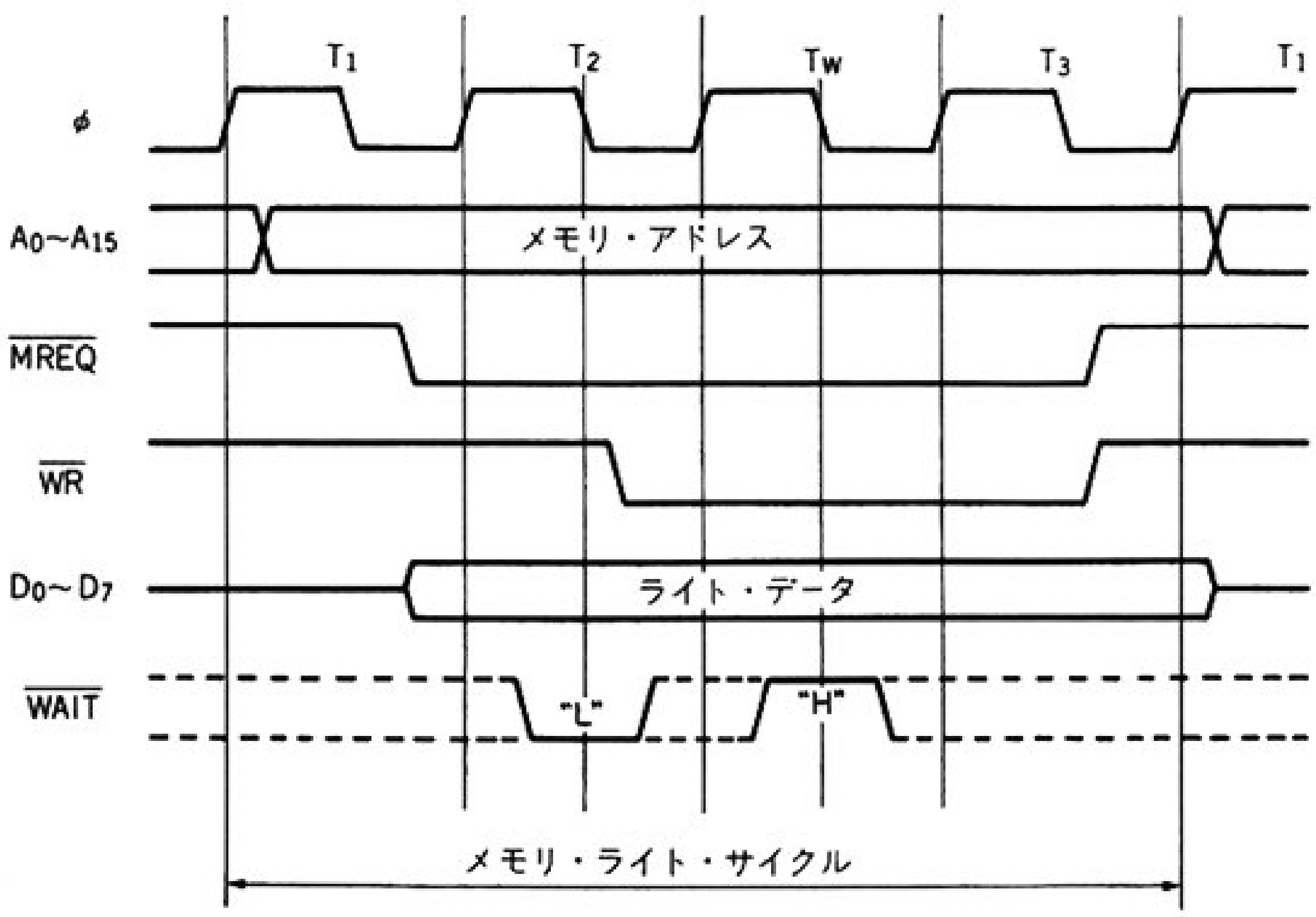
イト・ステート，T_Wと区別します．

T_W*ステートが挿入される理由は次のとおりです．I/Oリード/ライト・サイクルでは，T_2の始めに$\overline{IORQ}$信号が出力されます．$\overline{WAIT}$信号の状態が他のマシン・サイクルと同じく，T_2の立ち下がり時点にサンプルされるとすると，I/Oは$\overline{IORQ}$信号を受信してから非常にわずかな時間内に，アドレスをデコードし，かつ$\overline{WAIT}$信号をセットアップ時間を考慮して出力する必要があります．

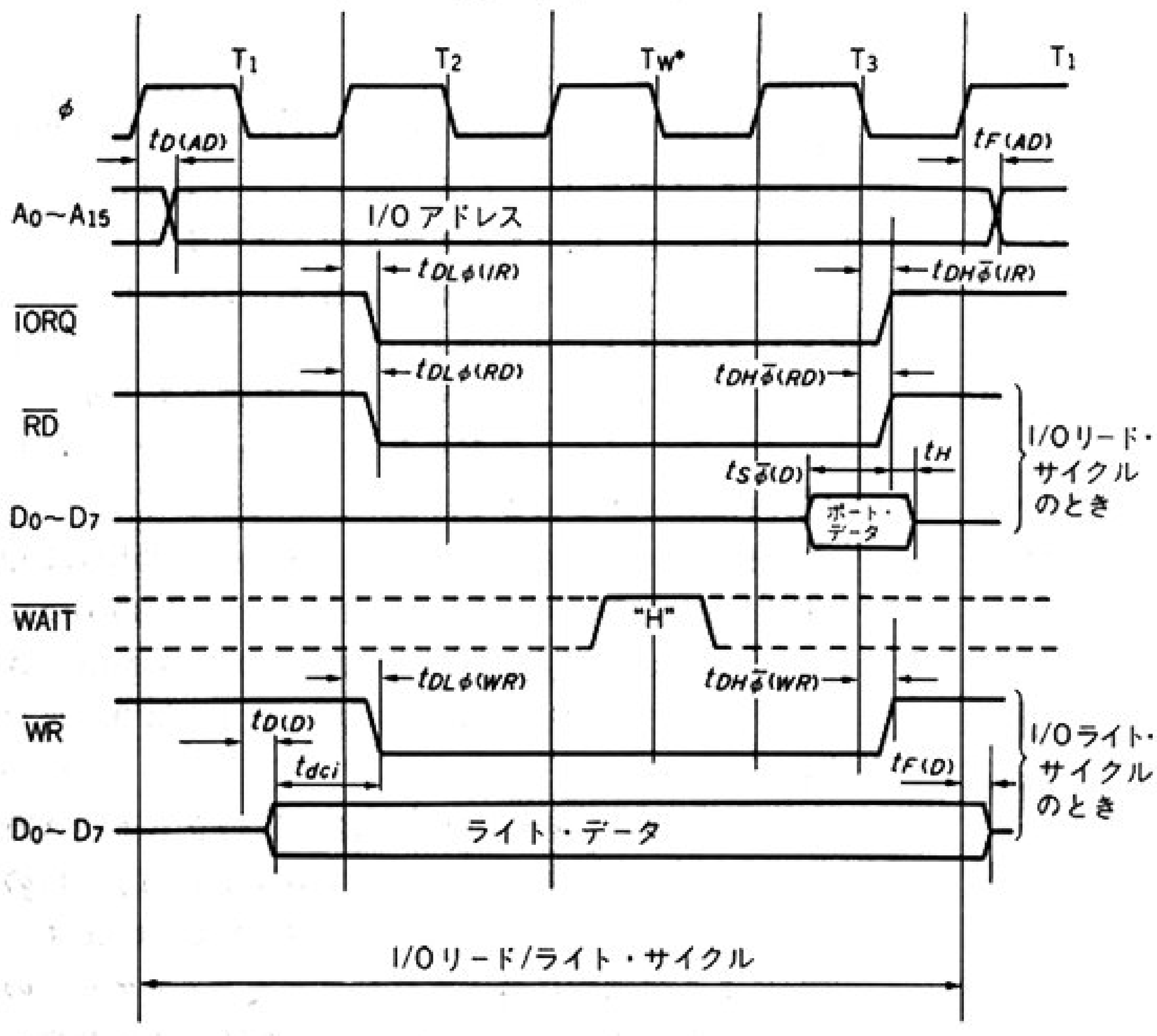
このため，T_2の後に強制的にT_W*を挿入し，$\overline{WAIT}$信号の状態のサンプル点を，T_W*のクロックの立ち下がりとすることにより，前述の時間的きびしさをなくすようにしているのです．

図24ではT_W*ステートのクロックの立ち下がり時点の$\overline{WAIT}$信号の状態は"H"なので，T_W*の次はT_3になっていますが，T_W*のクロックの立ち下がり時点の$\overline{WAIT}$信号の状態が"L"であると，T_W*の次にふつうのウェイト・ステートが挿入されます．図25に1ウェイト・ステートを挿入した，I/Oリード/ライト・サイクルのタイミングを示します．

〈図23〉 1ウェイト・ステートを挿入したメモリ・ライト・サイクル



〈図24〉 I/Oリード/ライト・サイクルのタイミング



割り込み認識サイクルのタイミング

割り込み認識サイクル(Interrupt Acknowledge Cycle)というのは，CPUが割り込み要求(INT＝"L")を受け付け，要求元から割り込みベクトルをリードするためのマシン・サイクルです．このマシン・サイクルを理解するには，Z 80の割り込みに関する知識が必須ですが，それについては後述の「Z 80の割り込み」の項をごらんください．

また，ここでは主としてZ 80のモード2割り込みのマシン・サイクルについて述べますが，モード0，およびモード1についても説明します．

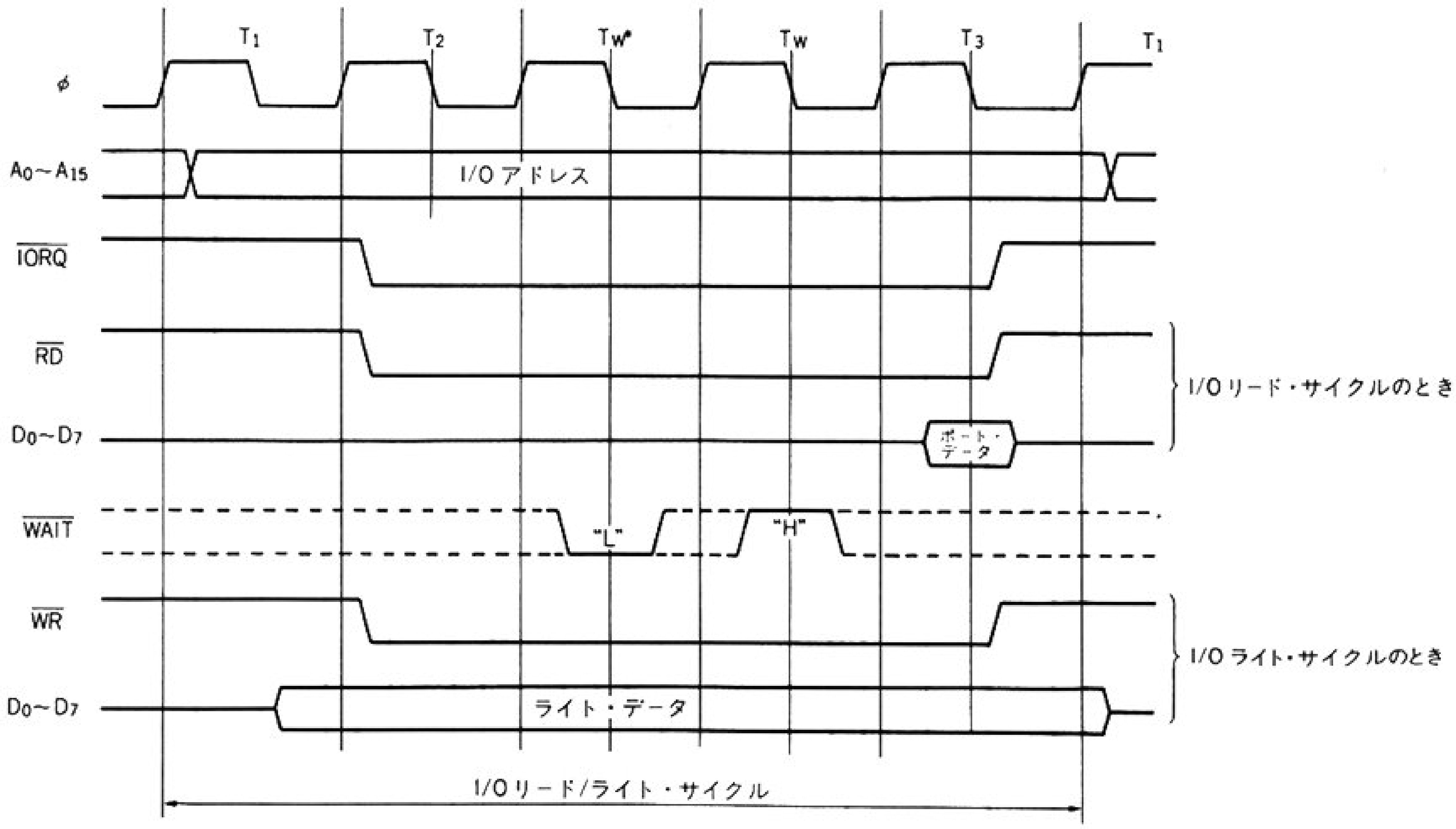
図26に割り込み認識サイクルのタイミングを示します．通常このマシン・サイクルは，割り込み要求を受け付け，割り込みベクトルをリードするまでのことをいいますが，図26には割り込みルーチンの先頭の命令をフェッチするところまで記述してあります．割り込み時における，Z 80の動作を完全に説明したかったからです．

この割り込み認識サイクルが起動されるのには，EI(イネーブル・インタラプト)命令により，CPU内部の割り込みイネーブル・フリップフロップがセットされており，かつ$\overline{BUSRQ}$信号が"H"である必要があります．

さて，このマシン・サイクルは次のようなステップからなります．

① 命令の最後のマシン・サイクルの最後のステートのクロックの立ち上がり時に，$\overline{INT}$信号の状態をサンプルす

〈図25〉 1ウェイト・ステートを挿入したI/Oライト・サイクルのタイミング



る．

② 割り込みベクトルをフェッチする．このサイクルは“スペシャルM 1”サイクルとも呼ばれ，$\overline{M1}$および$\overline{IORQ}$信号を共に“L”にすることにより，割り込み要求元から割り込みベクトルをリードします．

③ PCの上位バイトを(SP－1)にセーブする．

④ PCの下位バイトを(SP－2)にセーブする．

⑤ ②で読み取った割り込みベクトルから，割り込みルーチンのアドレス・テーブルをアクセスし，割り込みルーチンの先頭アドレスをメモリからリードする(下位バイト)．

⑥ 割り込みルーチンのアドレスの上位バイトをリードする．

⑦ ⑤，⑥で求めた割り込みルーチンのアドレスからOPコードをフェッチする．

前記②の割り込みベクトルをリードするサイクルには，自動的にウェイト・サイクルT_W*が二つ挿入されます．Z80の割り込みはデイジー・チェイン式の優先制御方式ですので，デイジー・チェインの末端まで信号が伝搬するための時間を，これらのT_W*により保証しています．

また，このサイクルは$\overline{RD}$信号の代わりに$\overline{IORQ}$が用いられている点を除くと，タイミング自体は通常のM 1サイクルとそっくりですので，“スペシャルM 1”サイクルと呼ばれます．

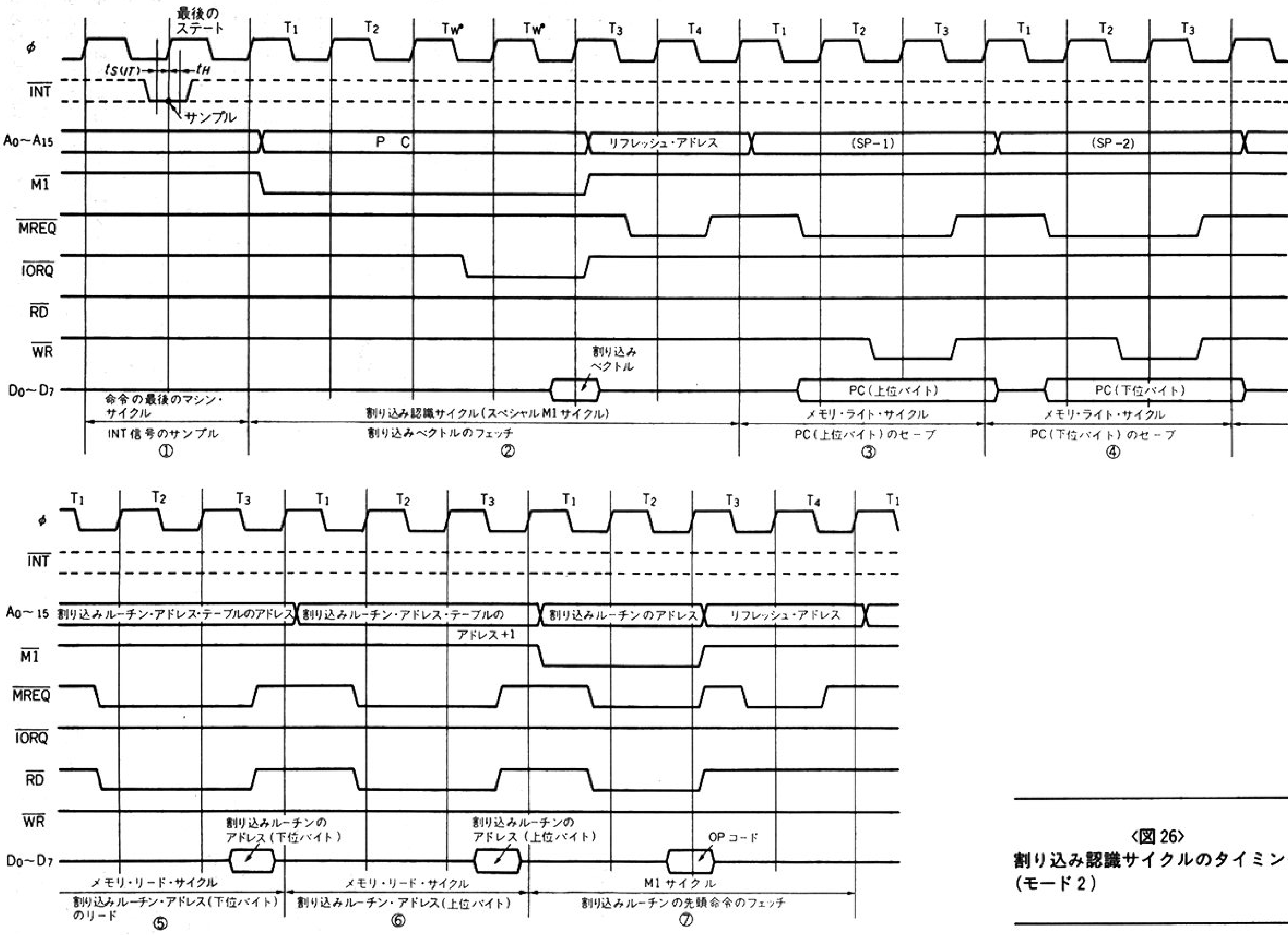
図26はモード2の割り込みのタイミングを示したものですが，モード0およびモード1ではどのようになるのか説明します．モード0では①～④の過程はモード2の場合とまったく同じですが，⑤および⑥の過程が省略されます．モード0では割り込みベクトルそのものが，割り込みルーチンのアドレスを示しているからです．

モード1の場合もタイミング自体はモード0とまったく同じですが，②で読み取った割り込みベクトルの値は無視されます．モード1では，ベクトルの値とは無関係に“RST 38 H”命令を実行するからです．

NMI認識サイクルのタイミング

NMIというのは，Non Maskable Interruptの略であり，“マクスできない割り込み”という意味です．$\overline{INT}$信号による割り込み要求は，EIおよびDIという命令により，割り込みを許可したり禁止したりすることができますが，このNMIは禁止することができず，常に割り込むことができる割り込み要求です．

図27にNMI認識サイクルのタイミングを示します．NMI認識サイクルは，$\overline{NMI}$信号を“L”にすることにより起動されます．CPUは$\overline{NMI}$信号が“L”になると，その立ち下がりにより，内部のNMIフリップフロップをセットします．NMIフリップフロップをセットするには，$\overline{NMI}$信号は最低，$T_{W(\overline{NMI})}$だけの時間，“L”である必要があります(28，29ページ)．



〈図26〉
割り込み認識サイクルのタイミング(モード2)

NMIフリップフロップの状態は，命令実行時の最後のマシン・サイクルの最後のステートの立ち上がり時点にサンプリングされ，このフリップフロップがセットされていると，次のマシン・サイクルは，NMI認識サイクルとなります．

このマシン・サイクルは通常のM1サイクルとまったく同じですが，CPUが読み取ったデータは何の意味もありません．したがってこのマシン・サイクルで意味があるのは，ダイナミックRAMのリフレッシュ動作だけです．次の2マシン・サイクルはメモリ・ライト・サイクルであり，これらのサイクルを用いて，CPUはPCの内容をスタックにセーブします．

PCの内容をセーブするマシン・サイクルの次には，M1サイクルが続き，アドレス・バス上には0066Hが出力されます．つまりCPUはNMIを検出すると，0066H番地の命令を実行します．0066H番地にはNMIに対処するプログラム(NMIルーチン)を書いておく必要があります．

CPU内部のNMIフリップフロップがリセットされるタイミングは，Zilog社のマニュアルからは明らかではありませんが，0066H番地の命令を実行するM1サイクルが終了した時点と考えるのが妥当です．

バス・リクエスト認識サイクルのタイミング

バス・リクエスト認識サイクル(Bus Request/Acknowledge Cycle)というのは，CPUが$\overline{BUSRQ}$信号を認識して，アドレス・バス，データ・バス，ならびにコントロール・バス($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, $\overline{WR}$, $\overline{RFSH}$)をハイ・インピーダンスにするためのマシン・サイクルです．

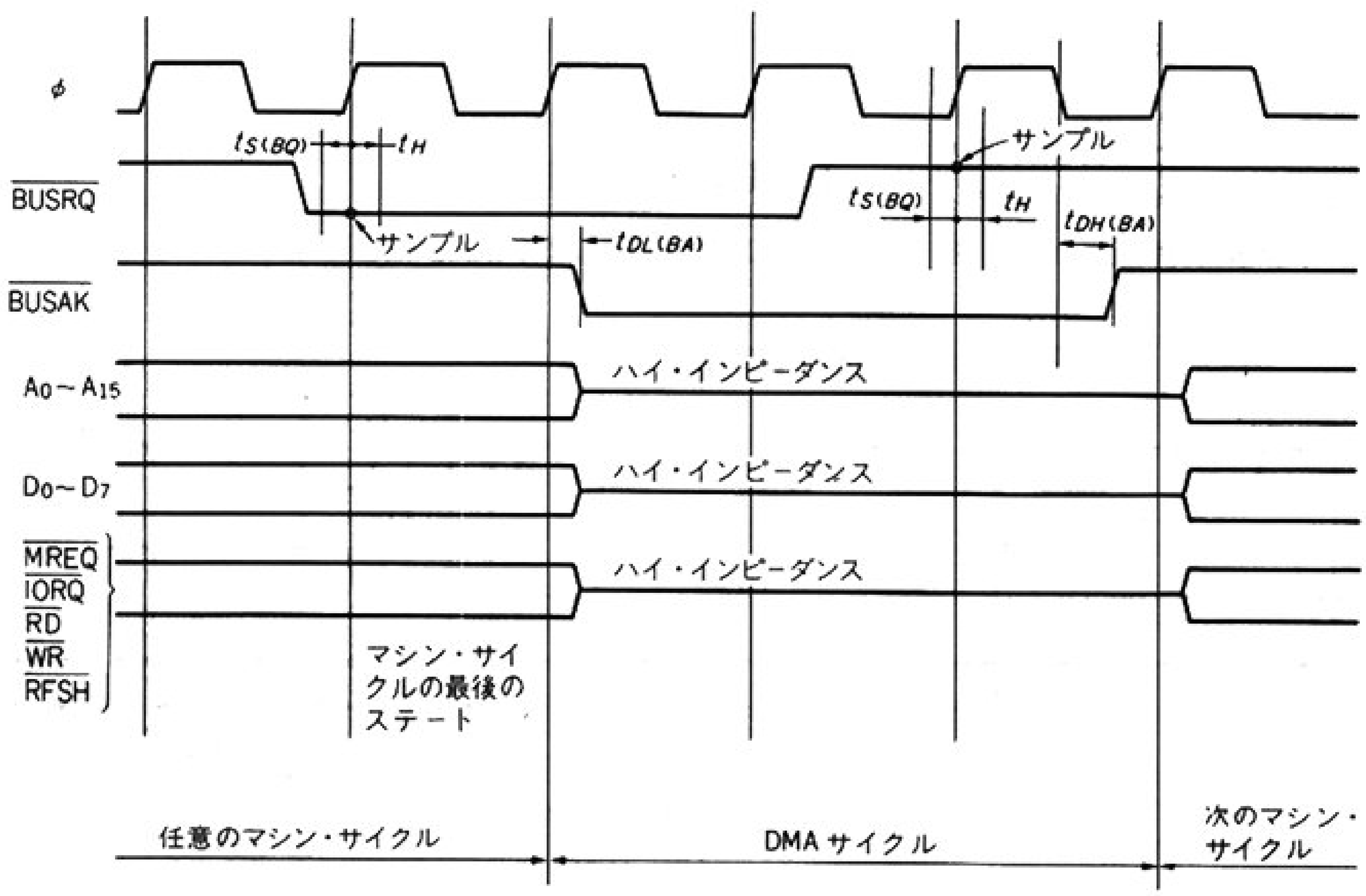
通常はCPUがこれらのバスの制御権を完全に握っているのですが，このマシン・サイクルを経て，バスの状態がハイ・インピーダンスになると，バスはCPUが接がっていないのとまったく同じ状態になります．したがってCPU以外の制御回路がバスをコントロールすることにより，CPUを介さないデータの転送をすることができます．

このように，CPUに対してバスの制御権を放棄させ，外部の制御回路によりバスをコントロールし，データの転送を行うことを，DMA(Direct Memory Access)データ転送といい，その制御を行う外部回路のことをDMAコントローラと呼びます．DMAデータ転送の目的は，CPUを介さずに高速でデータの転送を行うことです．

図28にバス・リクエスト認識サイクルのタイミングを示します．CPUは任意のマシン・サイクルの最後のステートのクロックの立ち上がり時点に$\overline{BUSRQ}$信号の状態をサンプリングします．ここで$\overline{BUSRQ}$信号の状態が"L"であると，次のクロックの立ち上がり時点にCPUは，$\overline{BUSAK}$信号を"L"にし，アドレス・バス，データ・バス，およびコントロール・バスの状態をハイ・インピーダンスにします．この状態をDMAサイクルと呼び，$\overline{BUSRQ}$信号が"L"である限り続きます．

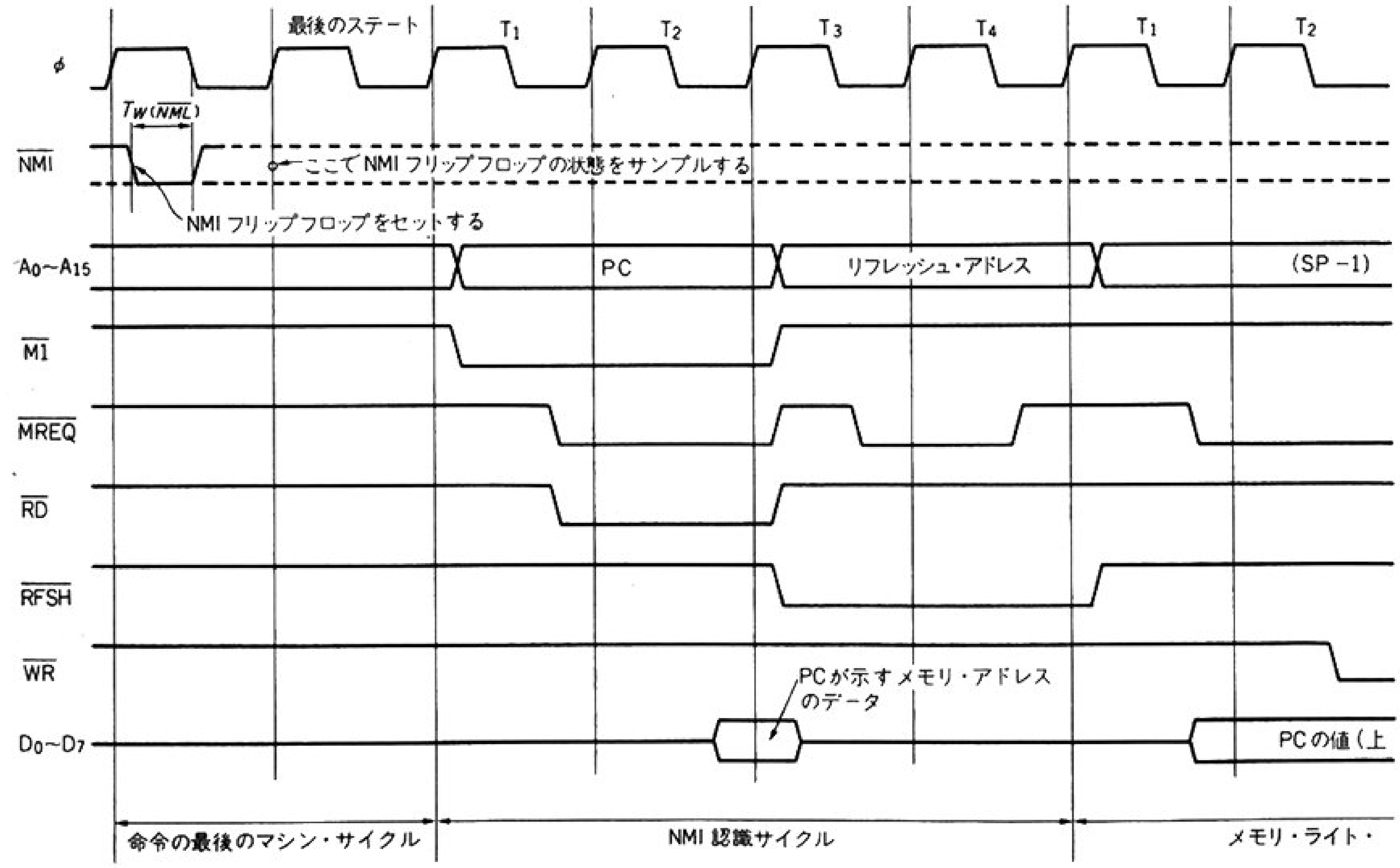
$\overline{BUSRQ}$信号のサンプリングには，セットアップ・タイム($t_{S(BQ)}$)とホールド・タイム(t_H)を満足する必要があります．

DMAサイクルを終結させるには，$\overline{BUSRQ}$信号を"H"に戻します．DMAサイクル中，CPUは各クロックの立ち上がり時点に$\overline{BURSQ}$信号の状態をサンプリングし，その状態が"H"であるのを検出すると，次のク



〈図28〉
バス・リクエスト認識サイクルのタイミング

〈図27〉 NMI認識サ



ロックの終わり頃に$\overline{BUSAK}$信号を"H"に戻し，その次のクロックの立ち上がり時点からバスの制御を再開します．

前項の割り込み要求の場合は，各命令の切れ目ごとに割り込み要求信号($\overline{INT}$，$\overline{NMI}$)のサンプリングを行っていましたが，このバス・リクエスト認識サイクルは，各マシン・サイクルの切れ目ごとに入ることができます．

したがってM1サイクルとメモリ・リード・サイクルを二つ用いるような命令の実行が，各マシン・サイクルの間にいくつかのDMAサイクルが挿入されて，行われるようなことも可能です．

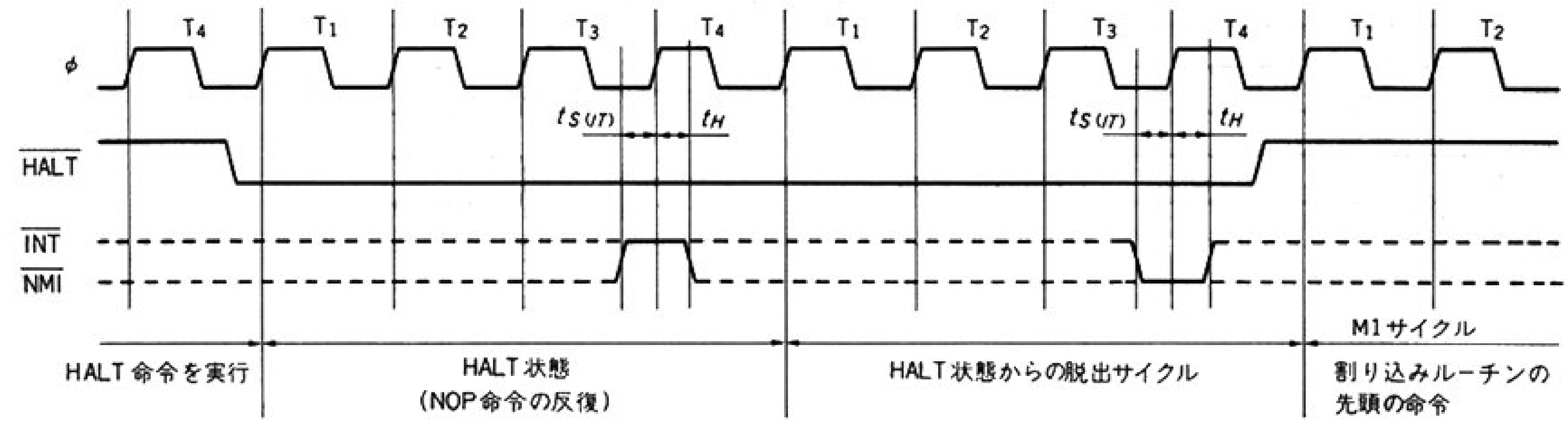
DMAサイクル中は，CPUはバスの制御をいっさい行わないので，ダイナミックRAMのリフレッシュ機能($\overline{RFSH}$)も停止します．

ですからZ80のリフレッシュ機能だけを用いてダイナミックRAMのリフレッシュ操作を行っているような場合，長時間DMAサイクルを続けると，メモリの内容が破壊される場合もあるので注意を要します．

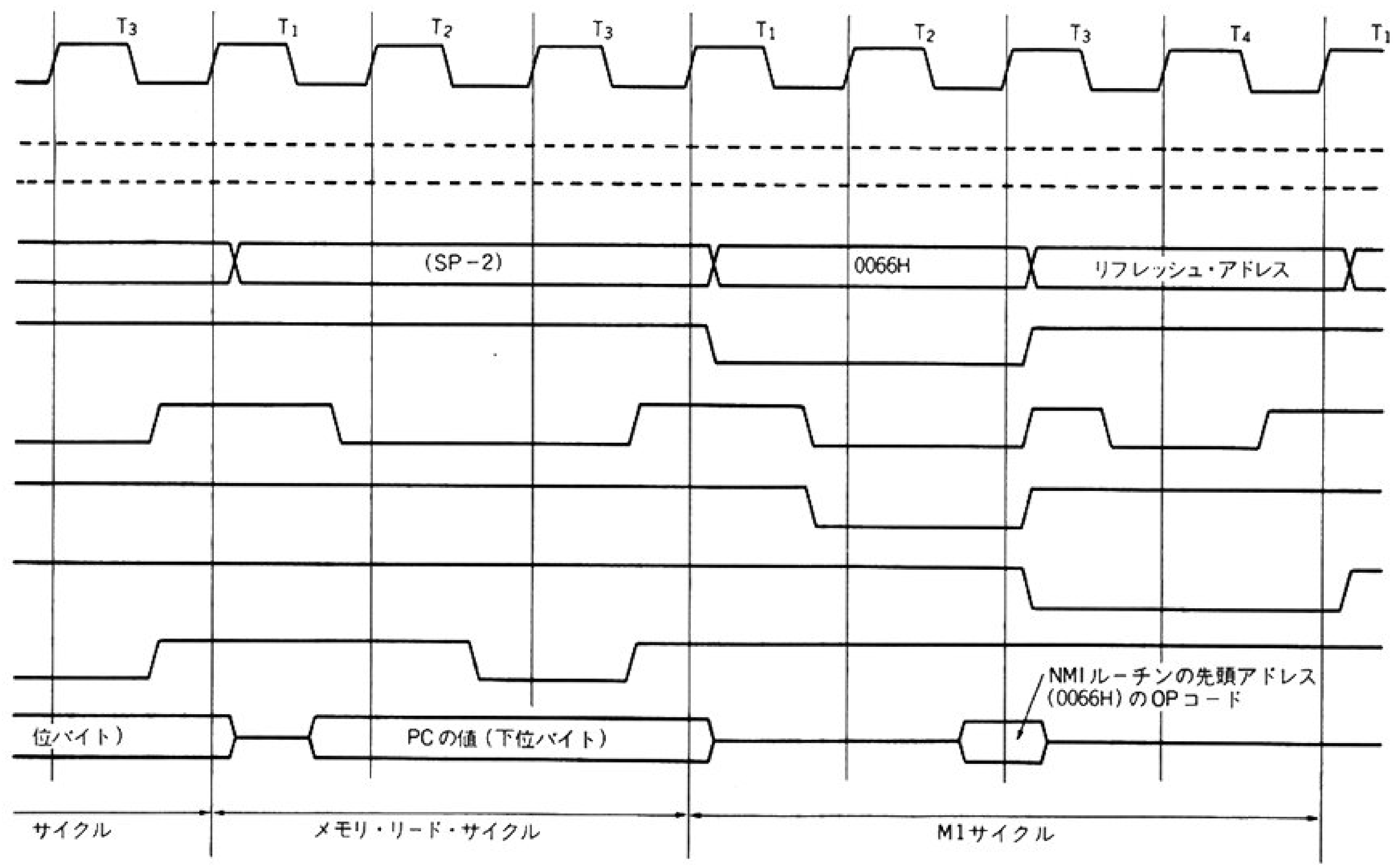
HALT状態からの脱出

Z80にはHALT(停止)という命令があります．CPU

〈図29〉 HALT状態からの脱出



イクルのタイミング



はこの命令を実行すると，最後のステート(T_4)の終わりに$\overline{\text{HALT}}$信号を“L”にします．そして次のマシン・サイクルから命令を先に進めることを中止して，M1サイクルだけからなるNOP命令を反復します．このM1サイクルは通常のM1サイクルとまったく同じように行われますが，データ・バスから読み取られたデータ(OPコード)は無視され，CPU内部にて強制的にNOP命令(00H)が挿入されます．この状態をHALT状態と呼びます．図29にHALT状態からの脱出のタイミングを示します．

このHALT状態から脱出できるのは，割り込み要求($\overline{\text{INT}}$または$\overline{\text{NMI}}$)が受け付けられたときだけです．HALT状態においても，割り込み要求はT_4のクロックの立ち上がり時点にサンプリングされ，$\overline{\text{INT}}$または$\overline{\text{NMI}}$信号が“L”であると，割り込みが発生し，割り込みルーチンにコントロールが移ります．割り込みが受け付けられた時点にHALT状態は解除され，T_4の終わり頃に$\overline{\text{HALT}}$信号は“H”に戻ります．

割り込みルーチンの実行が終り，その最後にRETIまたはRETN命令を実行することにより，HALT命令の次の命令へと進むことができます．HALT状態における割り込みの受け付けは，通常の場合とまったく同じように行なわれます．$\overline{\text{INT}}$信号による割り込みが受け付けられるには，EI命令により割り込み可の状態になっていなければなりません．

また，$\overline{\text{INT}}$信号と$\overline{\text{NMI}}$信号による割り込み要求が同時に発生した場合には，NMIのほうが優先度が高いので，NMIが先に受け付けられます．

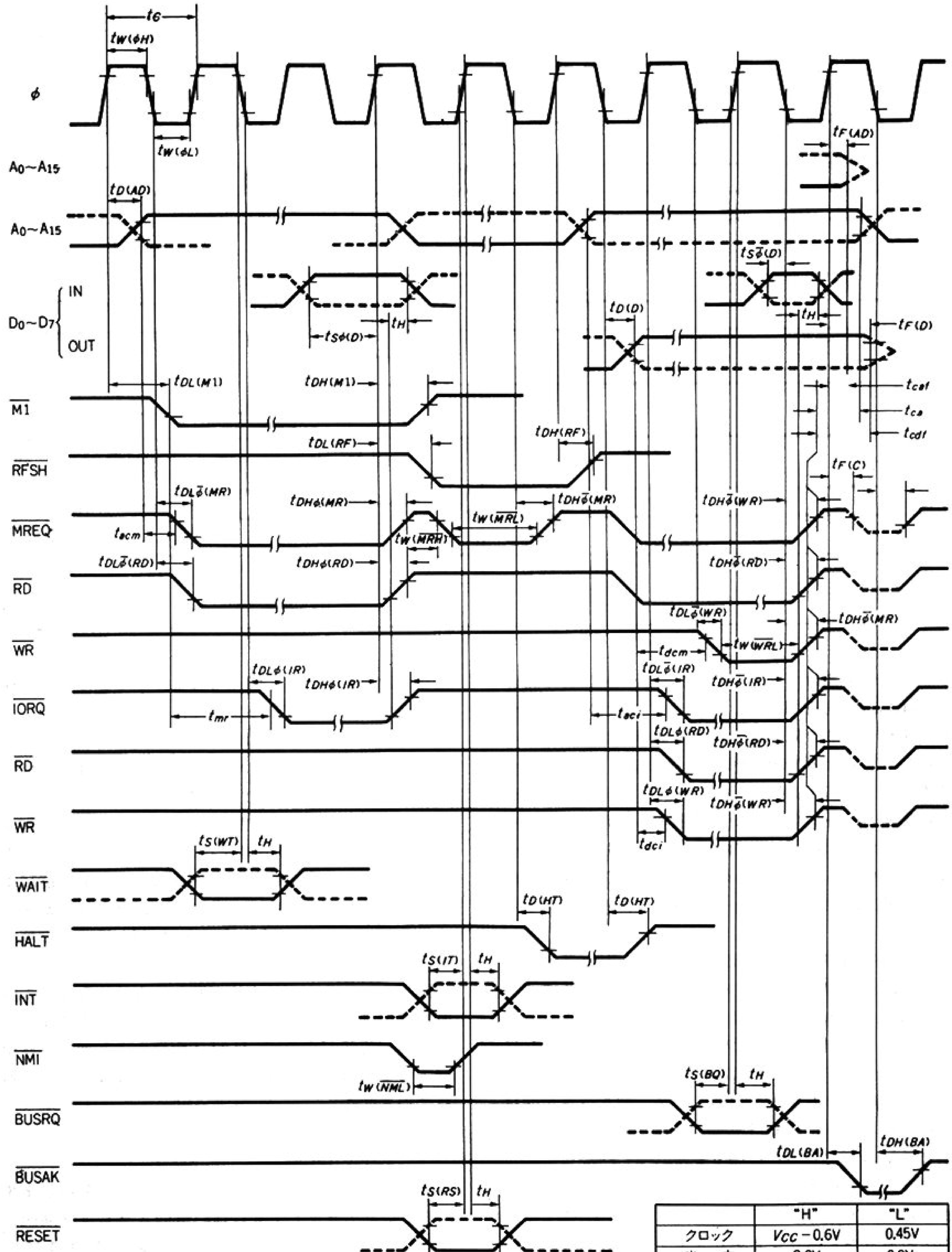
HALTという命令は“割り込みが発生するまで何もしないで待つ”という意味を持っていますが，割り込みというものは本来同時並行処理を行うことを可能にするためにあみ出されたくふうであり，“割り込みが発生するまで何もしない”ということは，割り込みの機能を否定しているようなものです．

最近のシステムでは高度の割り込み技法を使って同時並行処理を行い，CPUの能力をギリギリまで使っているのが現状ですので，HALT命令により割り込みが起こるまで待っているような使いかたはほとんど行われていません．

リセット・タイミング

図30にZ80CPUのリセット・タイミングを示します．Z80CPUをリセットするには，最低3クロック期間，$\overline{\text{RESET}}$信号を“L”にしておく必要があります．$\overline{\text{RESET}}$信号を受け付けると，Z80 CPUはアドレス・バス，およびデータ・バスをフローティング状態にし，

〈図 31〉 Z 80 CPUの一般的なタイミング・チャート



	"H"	"L"
クロック	$V_{CC}-0.6$V	0.45V
出　力	2.0V	0.8V
入　力	2.0V	0.8V
フローテイング	±0.5V	

〈図32〉 Z 80, -A, -B CPUのAC特性

(単位 ns)

信号名	記号	パラメータ	Z80 最小値	Z80 最大値	Z80A 最小値	Z80A 最大値	Z80B 最小値	Z80B 最大値
ϕ	t_C	クロック周期	400	式12	250	式12	165	式12
	$t_{W(\phi H)}$	クロック・パルス幅("H")	180		110		65	
	$t_{W(\phi L)}$	クロック・パルス幅("L")	180	2000	110	2000	65	2000
	t_r, t_f	クロックの立ち上がり/立ち下がり時間		30		30		20
$A_{0\sim15}$	$t_{D(AD)}$	アドレス出力の遅延時間		145		110		90
	$t_{F(AD)}$	フローティングになるまでの遅延時間		110		90		80
	t_{acm}	$\overline{MREQ}$に対するアドレス安定時間(メモリ・サイクル)	125(式1)		65(式1)		35(式1)	
	t_{aci}	$\overline{IORQ}$, $\overline{RD}$, $\overline{WR}$に対するアドレス安定時間(I/Oサイクル)	320(式2)		180(式2)		110(式2)	
	t_{ca}	$\overline{RD}$, $\overline{WR}$, $\overline{MREQ}$, $\overline{IORQ}$に対するアドレス保持時間	160(式3)		80(式3)		35(式3)	
	t_{caf}	クロックの立ち上がりからフローティングになるまでの遅延時間		110		90		80
$D_{0\sim7}$	$t_{D(D)}$	データ出力の遅延時間		230		150		130
	$t_{F(D)}$	フローティングまでの遅延時間		90		90		80
	$t_{S\phi(D)}$	M1サイクルにおけるクロックの立ち上がりに対するセットアップ・タイム	50		35			30
	$t_{S\overline{\phi}(D)}$	M2~M5サイクルにおけるクロックの立ち下がりに対するセットアップ・タイム	60		50		40	
	t_{dcm}	$\overline{WR}$に対するデータ安定時間(メモリ・サイクル)	190(式4)		80(式4)		25(式4)	
	t_{dci}	$\overline{WR}$に対するデータ安定時間(I/Oサイクル)	20(式5)		−10(式5)		−55(式5)	
	t_{cdf}	$\overline{WR}$が"H"になってからのデータ保持時間	120(式6)		60(式6)		30(式6)	
	t_H	ホールド・タイム		0		0		0
$\overline{MREQ}$	$t_{DL\overline{\phi}(MR)}$	クロックの立ち下がりから"L"になるまでの遅延時間		100		85		70
	$t_{DH\phi(MR)}$	クロックの立ち上がりから"H"になるまでの遅延時間		100		85		70
	$t_{DH\overline{\phi}(MR)}$	クロックの立ち下がりから"H"になるまでの遅延時間		100		85		70
	$t_{W(\overline{MRL})}$	パルス幅("L")	360(式7)		220(式7)		135(式7)	
	$t_{W(\overline{MRH})}$	パルス幅("H")	170(式8)		110(式8)		65(式8)	
$\overline{IORQ}$	$t_{DL\phi(IR)}$	クロックの立ち上がりから"L"になるまでの遅延時間		90		75		65
	$t_{DL\overline{\phi}(IR)}$	クロックの立ち下がりから"L"になるまでの遅延時間		110		85		70
	$t_{DH\phi(IR)}$	クロックの立ち上がりから"H"になるまでの遅延時間		100		85		70
	$t_{DH\overline{\phi}(IR)}$	クロックの立ち下がりから"H"になるまでの遅延時間		110		85		70
$\overline{RD}$	$t_{DL\phi(RD)}$	クロックの立ち上がりから"L"になるまでの遅延時間		100		85		70
	$t_{DL\overline{\phi}(RD)}$	クロックの立ち下がりから"L"になるまでの遅延時間		130		95		80
	$t_{DH\phi(RD)}$	クロックの立ち上がりから"H"になるまでの遅延時間		100		85		70
	$t_{DH\overline{\phi}(RD)}$	クロックの立ち下がりから"H"になるまでの遅延時間		110		85		70
$\overline{WR}$	$t_{DL\phi(WR)}$	クロックの立ち上がりから"L"になるまでの遅延時間		80		65		60
	$t_{DL\overline{\phi}(WR)}$	クロックの立ち下がりから"L"になるまでの遅延時間		90		80		70
	$t_{DH\overline{\phi}(WR)}$	クロックの立ち下がりから"H"になるまでの遅延時間		100		80		70
	$t_{W(\overline{WRL})}$	$\overline{WR}$のパルス幅	360(式9)		220(式9)			135(式9)
$\overline{M1}$	$t_{DL(M1)}$	クロックの立ち上がりから"L"になるまでの遅延時間		130		100		80
	$t_{DH(M1)}$	クロックの立ち上がりから"H"になるまでの遅延時間		130		100		80
$\overline{RFSH}$	$t_{DL(RF)}$	クロックの立ち上がりから"L"になるまでの遅延時間		180		130		110
	$t_{DH(RF)}$	クロックの立ち上がりから"H"になるまでの遅延時間		150		120		100
$\overline{WAIT}$	$t_{S(WT)}$	クロックの立ち下がりに対するセットアップ・タイム	70		70		60	
$\overline{HALT}$	$t_{D(HT)}$	クロックの立ち下がりから"L"または"H"になるまでの遅延時間		300		300		260
$\overline{INT}$	$t_{S(IT)}$	クロックの立ち上がりに対する, セットアップ・タイム	80		80		70	
$\overline{NMI}$	$t_{W(\overline{NML})}$	$\overline{NMI}$のパルス幅	80		80		70	
$\overline{BUSRQ}$	$t_{S(BQ)}$	クロックの立ち上がりに対する, セットアップ・タイム	80		50		50	
$\overline{BUSAK}$	$t_{DL(BA)}$	クロックの立ち上がりから"L"になるまでの遅延時間		120		100		90
	$t_{DH(BA)}$	クロックの立ち下がりから"H"になるまでの遅延時間		110		100		90
$\overline{RESET}$	$t_{S(RS)}$	クロックの立ち上がりに対するセットアップ・タイム	90		60		60	
	$t_{F(C)}$	$\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, $\overline{WR}$がフローティングになるまでの遅延時間		110		90		80
	t_{mr}	$\overline{IORQ}$が"L"になる前に, $\overline{M1}$が"L"になる時間(割り込み認識サイクル)	920(式10)		565(式10)		365(式10)	

・式番号が記入してあるものは, t_C, $t_{W(\phi H)}$および$t_{W(\phi L)}$が最小値で $t_r=t_f=20$ns とした場合, その他の場合には次頁の表により計算する。

・Z80CPUはスタティック動作ができるように設計されているが, $t_{W(\phi H)}$の値が200μsを越えるような場合の動作は保証されない。

〈図 32〉 つづき

式番号	記 号	Z80	Z80A	Z80B
1	t_{acm}	$t_{W(\phi H)}+t_f-75$	$t_{W(\phi H)}+t_f-65$	$t_{W(\phi H)}+t_f-50$
2	t_{aci}	t_C-80	t_C-70	t_C-55
3	t_{ca}	$t_{W(\phi L)}+t_r-40$	$t_{W(\phi L)}+t_r-50$	$t_{W(\phi L)}+t_r-50$
4	t_{dcm}	t_C-210	t_C-170	t_C-140
5	t_{dci}	$t_{W(\phi L)}+t_r-180$	$t_{W(\phi L)}+t_r-140$	$t_{W(\phi L)}+t_r-140$
6	t_{cdf}	$t_{W(\phi L)}+t_r-80$	$t_{W(\phi L)}+t_r-70$	$t_{W(\phi L)}+t_r-55$
7	$t_{W(\overline{MRL})}$	t_C-40	t_C-30	t_C-30
8	$t_{W(\overline{MRH})}$	$t_{W(\phi H)}+t_f-30$	$t_{W(\phi H)}+t_f-20$	$t_{W(\phi H)}+t_f-20$
9	$t_{W(\overline{WRL})}$	t_C-40	t_C-30	t_C-30
10	t_{mr}	$2t_C+t_{W(\phi H)}+t_f-80$	$2t_C+t_{W(\phi H)}+t_f-65$	$2t_C+t_{W(\phi H)}+t_f-50$
12	t_C	$t_{W(\phi H)}+t_{W(\phi L)}+t_r+t_f$		

〈図 33〉
Z 80 CPUのDC特性と容量特性

■DC特性

記号	パラメータ	最小値	最大値	単位	テスト条件
V_{ILC}	クロック入力の“L”レベル	0.3	0.45	V	
V_{IHC}	クロック入力の“H”レベル	$V_{CC}-0.6$	$V_{CC}+0.3$	V	
V_{IL}	入力“L”レベル	−0.3	0.8	V	
V_{IH}	入力“H”レベル	2.0	V_{CC}	V	
V_{OL}	出力“L”レベル		0.4	V	$I_{OL}=1.8\,mA$
V_{OH}	出力“H”レベル	2.4			$I_{OH}=-250\,\mu A$
I_{CC}	電源電流 Z 80 Z 80 A Z 80 B		 150 200 200	mA	
I_{LI}	入力漏れ電流		10	μA	$V_{IN}=0\sim V_{CC}$
I_{LEAK}	フローティング状態における3ステート出力の漏れ電流	−10	10^3	μA	$V_{OUT}=0.4\sim V_{CC}$

■容量特性(25℃, 1 MHz)

記号	パラメータ	最小値	最大値	単位	注
C_{CLK}	クロック容量		35	pF	測定に関与しないピンはクラウンド
C_{IN}	入力容量		5	pF	測定に関与しないピンはクラウンド
C_{OUT}	出力容量		10	pF	測定に関与しないピンはクラウンド

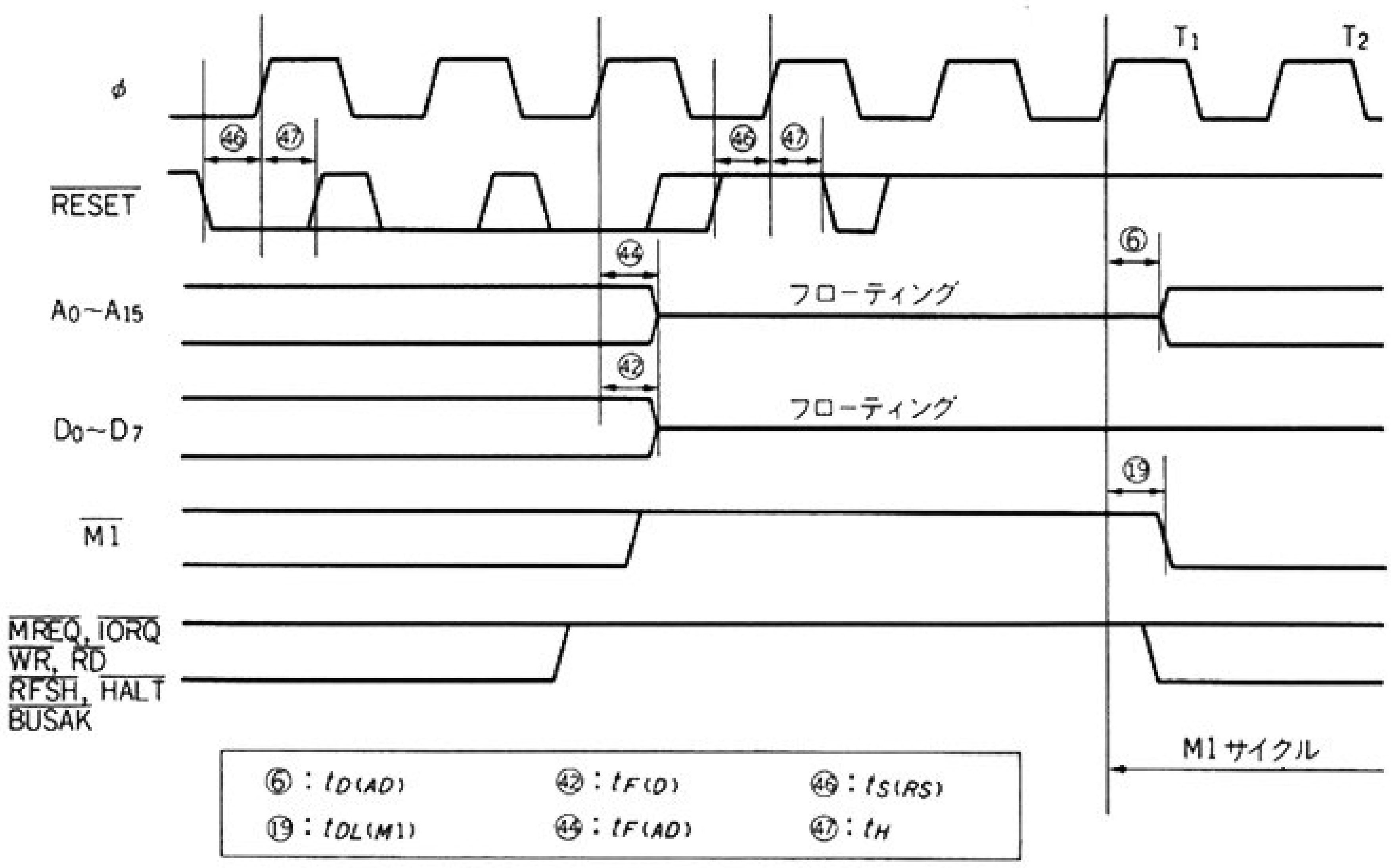
制御信号($\overline{MREQ}$, $\overline{IORQ}$など)を“H”にします. $\overline{RESET}$信号が“H”になると, 3クロック目の立ち上がり時点からM1サイクルが開始され, 0000H番地の命令の実行が始まります(33ページ).

$\overline{RESET}$信号に関して注意すべきことは, この信号についてもセットアップ・タイムとホールド・タイムがあることです. しかしながら通常は十分に長い$\overline{RESET}$信号を印加するので, これらに注意を払う必要はほとんどありません.

以上でZ80 CPUのタイミングの項を終わりますが, 図31～図33に, Z80(2.5MHz), Z80A(4MHz), Z80B(6 MHz)の一般的なタイミング・チャート, ならびにAC特性, DC特性を示します.

本項で示したタイミング・チャートには, 各マシン・サイクルにおいて着目すべき時間的定数の記号が示されていますが, それらはここで示す一般的なタイミング・チャートからとったものです. 実際の値については, 図32のAC特性をごらんください.

筆者の経験からすると, 個々のマシン・サイクルを



〈図30〉
リセット・タイミング

検討する際には，図31のタイミング・チャートから，時間的定数を求めてくるのは，相当骨の折れることです．したがって本項のタイミング・チャートに示された，時間的定数を示す記号はきっと役に立つはずです．

Z80 CPUの割り込み

Z80には2種類の割り込み要求，

① $\overline{INT}$信号によるマスク(Mask——禁止)可能な割り込み(以降INT割り込み)

② $\overline{NMI}$信号によるマスク不可能な割り込み(以降NMI)

があります．NMIはINT割り込みより優先度が高く，両者が同時に発生した場合，NMIのほうが先に受け付けられます．前項ではこれらの割り込みのタイミングについて説明をしましたが，ここではこれらをプログラミングする立場から説明します．

NMI(Non Maskable Interrupt)

前述したようにNMIは文字どおり，「マスクすることができない割り込み」であり，常に受け付けられます．したがってINT割り込みの処理ルーチンの実行中であろうと，NMI割り込みルーチンの実行中であろうと，$\overline{NMI}$信号の立ち下がりが検出されさえすれば，常にNMIが発生します．

NMIはシステムの非常事態，たとえば電源異常，メモリ・パリティ・エラー発生などの状態を，一早くプログラムに伝えるためにあります．

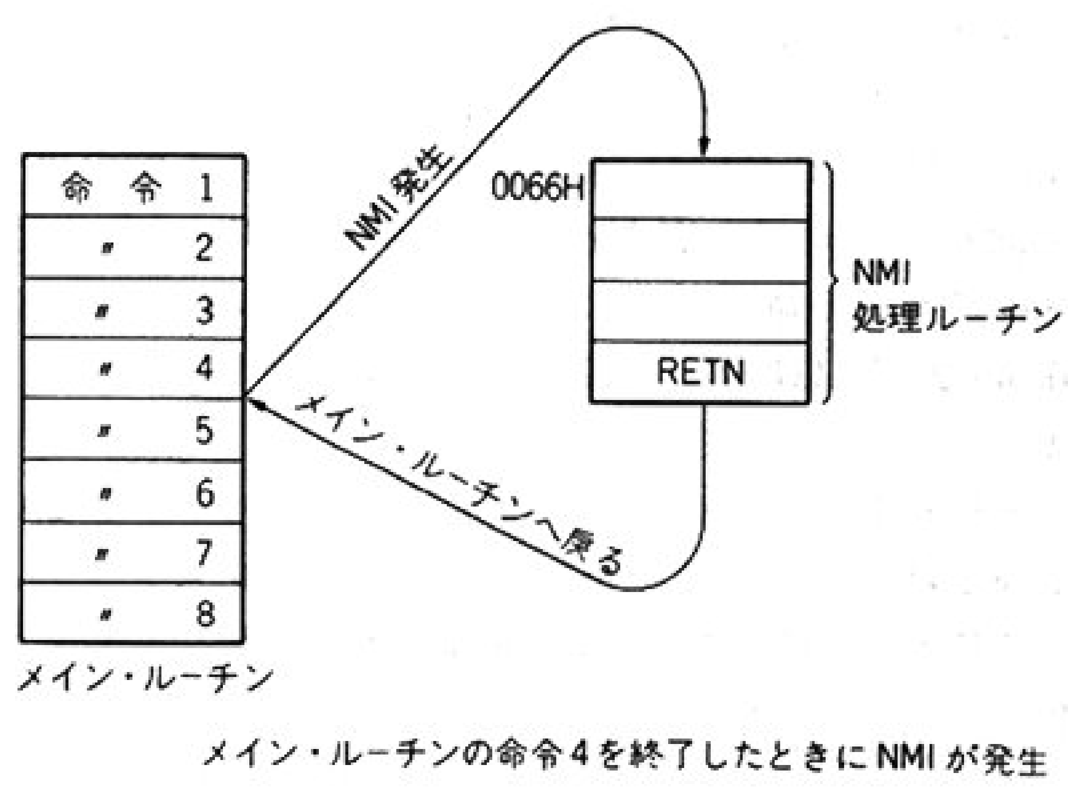
Z80はNMIを認識すると，その時点のPCの内容をスタックにセーブした後に，0066H番地の命令にジャンプします．したがってNMI処理ルーチンの先頭アドレスは，常に0066H番地にしておく必要があります．そしてNMI処理ルーチンの最後には，RETN命令を実行し，NMIが発生する前のルーチンに戻ります．図34にNMI発生時のプログラムの動きを示します．

Z80を使っている人の中には「NMIはRETN命令を実行しなければ，再度発生しない」と考えている人がたまにはいますが，これが誤りであることを示す実験をやってみました．

図35にこの実験を行うための簡単なロジックを示します．このロジックはOUT (1EH)，A命令を出すと，$\overline{IORQ}$の立ち下がり時点にワンショットをトリガします．このワンショットは約100 nsのパルスを出力し，このパルスはZ80 CPUの$\overline{NMI}$ピンに接続されています．

前記のロジックに対して，Prog❶に示すプログラム

〈図34〉NMI発生時のプログラムの動き

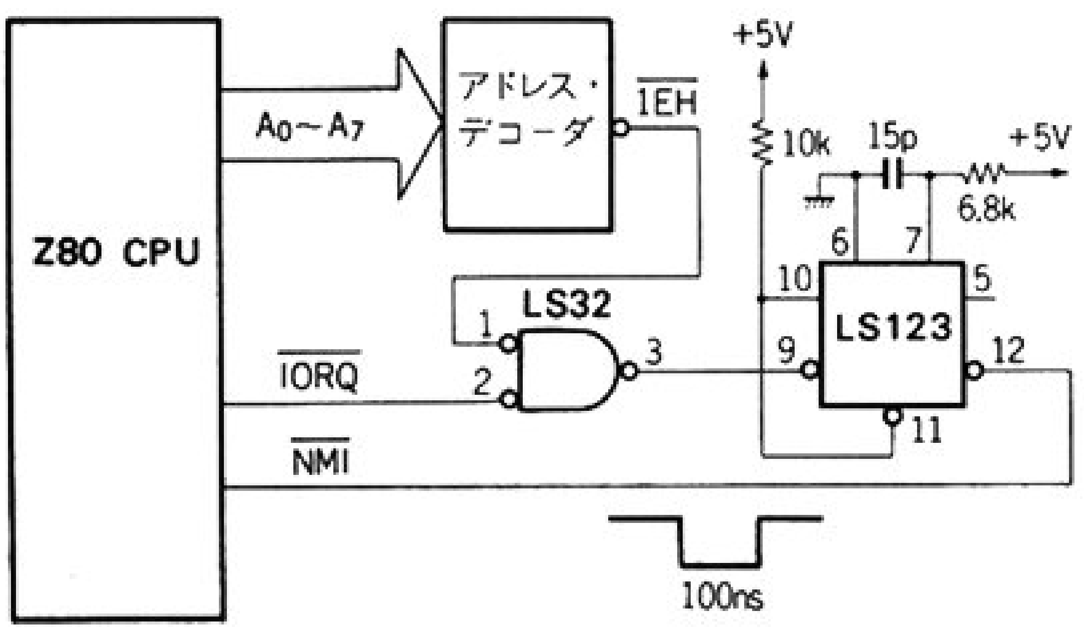


メイン・ルーチンの命令4を終了したときにNMIが発生

〈Prog❶〉 NMIテスト・プログラム

```
                              ;***********************************************************
                              ;****              NMI TEST                            ****
                              ;***********************************************************
0000    31 0000                        LD       SP,STACK        ;SET STACK POINTER
0003    D3 1E                          OUT      (1EH),A         ;TRIGGER ONE SHOT
                              ;
                              ;
                                       ORG      066H
0066    31 0000               NMI:     LD       SP,STACK        ;RESET STACK POINTER
0069    D3 1E                          OUT      (1EH),A         ;TRIGGER ONE SHOT
                              ;
0000                          STACK    EQU      0000H           ;
                              ;
                                       END
```

〈図35〉 NMIテストのためのロジック



を動かしてみます。このプログラムを0000H番地からスタートさせると，まずOUT (1 EH), A命令が実行されます。この命令により出力される$\overline{\text{IORQ}}$信号の立ち下がり時点にワンショットがトリガされ，$\overline{\text{NMI}}$信号がZ 80に入力されます。

Z 80はこの$\overline{\text{NMI}}$信号の立ち下がり時に，内部のNMIフリップフロップをセットし，OUT (1 EH), A命令の最後のマシン・サイクル(I/Oライト・サイクル)の最後のステートのクロックの立ち上がり時点に，NMIフリップフロップがセットされていることを認識し，NMIを発生します。

NMIが発生すると0066H番地に飛んできます。このプログラムでは，0066H番地でスタック・ポインタをリセットしたのち，再びOUT (1EH), A命令を実行します。このOUT (1EH), A命令が実行されると前の説明と同じ過程が繰り返され，再度NMIが発生し，以後0066H番地の命令(LD SP, STACK)と0069H番地の命令(OUT (1EH), A)を永久に繰り返すようになります。

0066H番地でスタック・ポインタをリセットしている理由は，こうしておかないとメモリ中がスタックとして使われ，結果的にはProg❶自体を破壊してしまうからです。

INT割り込み

INT割り込みはマスク可能な割り込みであり，EI (Enable Interrupt)命令により割り込みを許可し，DI (Disable Interrupt)命令により，割り込みを禁止することができます。またNMIが発生するとINT割り込みは強制的に禁止されます。さらにZ 80のINT割り込みには，モード0～モード2の3種類の割り込みモードがあります。

ここではまずINT割り込みの許可/禁止(イネーブル/ディスエーブル)のメカニズムについて説明し，次に各割り込みモードについて説明します。

INT割り込みの イネーブル/ディスエーブル

Z 80におけるINT割り込みのイネーブル/ディスエーブル制御は，二つの割り込みイネーブル・フリップフロップ，IFF_1およびIFF_2により行います。図36にこれら二つのフリップフロップの状態が変化する要因と，フリップフロップの状態を示します。

Z 80がリセットされると，両フリップフロップともリセット("0")されます。EI命令が実行されると，両フ

〈図36〉 IFF_1とIFF_2の動作

要　因	IFF_1	IFF_2	コメント
リセット	0	0	
DI命令	0	0	
EI命令	1	1	
INT割り込み受け付け	0	0	
NMI受付け	0	X	$IFF_1=0$
RETN命令	IFF_2	X	$IFF_2 \rightarrow IFF_1$
LD A, I命令	X	X	IFF_2→パリティ・フラグ
LD A, R命令	X	X	IFF_2→パリティ・フラグ

X：変化しない

リップフロップはセット("1")され，DI命令が実行されると両方ともリセットされます．

また，INT割り込みが受け付けられると，IFF_1，IFF_2はともにリセットされます．そしてINT割り込みが許可されるのは，両フリップフロップがともにセットされているときだけです．

これまでの話からすると，「何だ，フリップフロップ1個で十分じゃないか」と思われますが，NMIが発生しない限り，その答は"イエス"です．ところがZ80ではNMIが発生すると，INT割り込みを強制的に禁止すると述べましたが，その制御に一役買うのがIFF_2です．

NMIを受け付けると，Z80はIFF_1をリセットします．IFF_2の状態はそのままであり，NMI直前のIFF_1の状態を示しています．IFF_1がリセットされたので，以後INT割り込みは禁止されます．そしてこの状態はNMI処理ルーチン内でEI命令を実行するか，RETN命令を実行するまで続きます．

RETN命令を実行すると，IFF_2にセーブされたIFF_1の状態がIFF_1に戻されます．したがって，NMIが発生する直前にEI状態であったとすれば，RETN命令によってその状態が回復されます．つまり，Z80ではNMIが発生する直前のEI/DI状態をセーブしておき，RETN命令によってNMIが起きる直前の状態に戻す動作が行われます．

たとえNMIが発生したところで，INT割り込みのイネーブル/ディスエーブル状態を変えたくない，というようなユーザーのために，IFF_2の状態をチェックする手段があります．LD A，IまたはLD A，R命令を実行すると，IFF_2の状態をステータス・フラグ中のパリティ・フラグ(V)にセットしてくれます．したがって，このフラグを調べて，もしセットされていれば，EI命令を出すことにより，INT割り込みのイネーブル状態を継続することができます．

なおEI命令について付言しておくと，実際には割り込みがイネーブルされるのは，EI命令の次の命令が実行された後です．割り込みルーチンの最後には通常，EI命令とRETI命令が並んでいますが，割り込みがイネーブルされるのは，RETI命令の実行後であるということです．

またEI命令とDI命令を連続して実行した場合，割り込みは発生しません．EI命令の次の命令が終わった後に割り込みはイネーブルされるはずですが，DI命令によりディスエーブルされてしまうからです．短期間だけ割り込みをイネーブルしたいときには，EI，NOP，DIの順に命令を実行すれば，NOP命令の後に割り込みを発生させることができます．

INT割り込みのモード

Z80にはモード0～モード2という，3種類の割り込みモードがあります．ここではこれらの各割り込みモードにおけるZ80の動作を説明します．

■モード0割り込み

モード0は8080Aの割り込みとまったく同じ割り込みモードです．Z80はリセットされると，このモード0になり，8080Aのソフトウェアとの互換性を保ちます．Z80が他の割り込みモード(モード1またはモード2)にあるときには，IM 0という命令を実行すると，モード0になります．

モード0の割り込みは，割り込み認識サイクルにおいて，割り込み源から任意の命令を読み込んで，それを実行するモードです．割り込み認識サイクルのタイミングの項では，割り込みモードをモード2であると仮定し，割り込み源は割り込みベクトルをCPUに送ると説明しましたが，このモード0では割り込み源は任意の命令をCPUに送ることができます．割り込み認識サイクルは，M1サイクルとほとんど変わらないのでこのようなことが可能です．

しかしながら，このモード0における割り込み認識サイクルでは，PCを自動的にスタック上にセーブする動作は行われませんので，割り込まれたプログラムのPCをスタック上にセーブできるような命令をCPUに送ってやる必要があります．したがって，割り込み源はRST n命令，またはCALL nn 命令をCPUに送るのがふつうです．

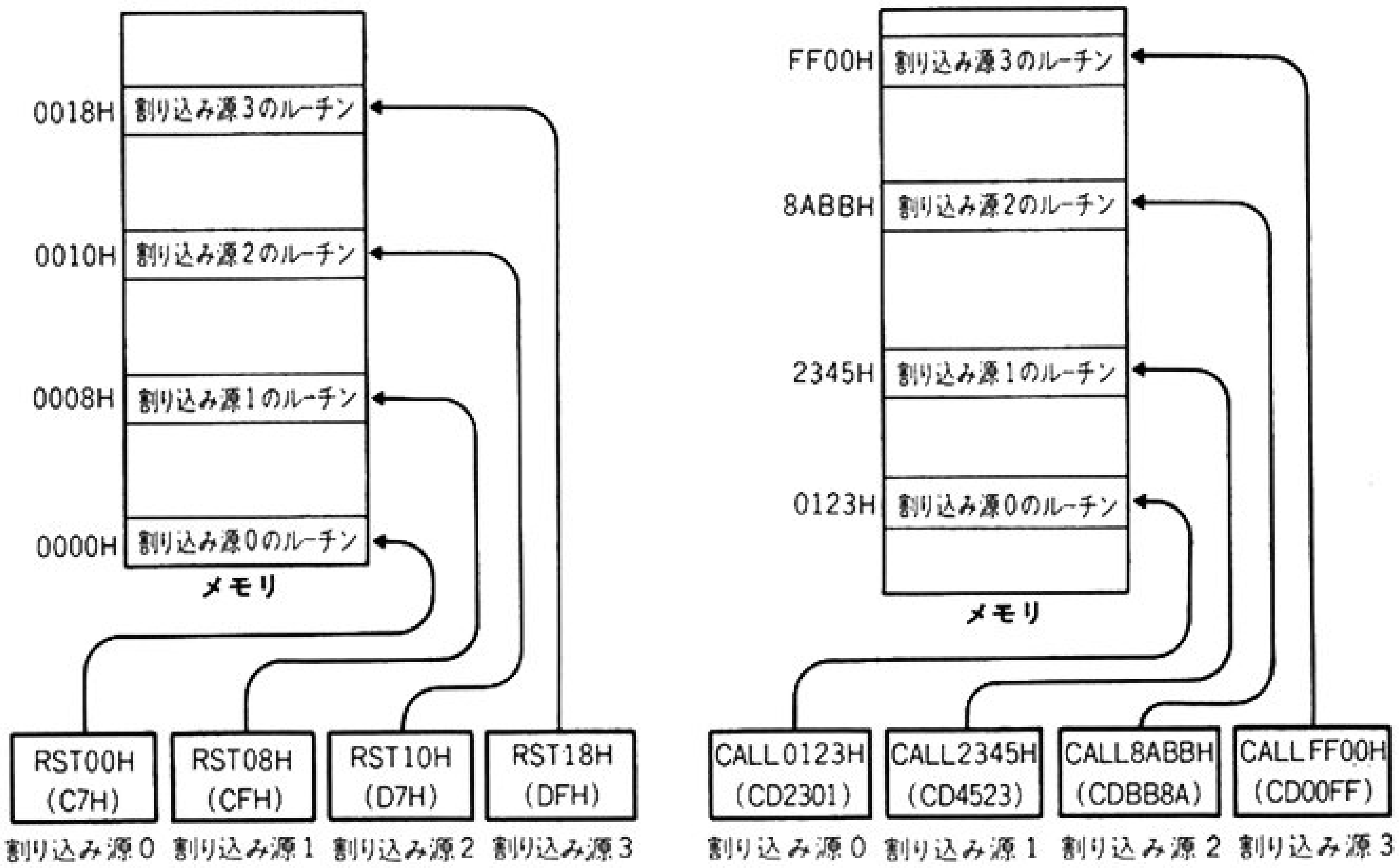
RST n命令は，1バイト命令であり，OPコード中の3ビットにより，RST 00H～RST 38H命令に変化しますので，比較的簡単な外部回路を設けるだけで，8種類の割り込みを識別できます．8種類の割り込みだけでは物足りない場合には，3バイトからなるCALL nn 命令をCPUに送ります．

この場合，原理的には64K種類の割り込みを識別できますが，3バイトの命令を送る必要があるので，外部回路は複雑になります．8080A用に作られた，初期のインタラプト・コントローラ，8214はRST命令を発生し，後の8259はCALL命令を発生します．図37に以上の説明をまとめた図を示します．

■モード1割り込み

IM 1という命令を実行すると，割り込みモードは，モード1になります．このモードでは，Z80は割り込み認識サイクル時に割り込み源から読み取った命令を無視し，内部で強制的にRST 38H命令を挿入します．したがって割り込み源は，割り込み認識サイクル時に，

〈図37〉
モード0割り込み



(a) RST 命令を送る場合　(b) CALL 命令を送る場合

データ・バス上に何も乗せる必要がなく，割り込みが受け付けられると，常に0038H番地へ飛びます．

8080Aにおいても，データ・バスをプル・アップしておき，割り込み認識サイクル時に割り込み源がデータ・バス上に何も乗せなければ，CPUは命令FFH (RST 38H)を実行するので，このモード1も8080Aとコンパチブルな割り込みモードということができます．

■モード2割り込み

モード2割り込みはZ80の特徴の一つに数えられるものであり，これこそZ80本命の割り込みモードである，ということができます．このモード2割り込みの特徴は，次のとおりです．

①1バイトの割り込みベクトルを送るだけで，128種類の割り込みを識別できる．

②割り込みルーチンのアドレス・テーブルをメモリ上のどこにでも置くことができる．

Z80をこのモード2にするにはIM 2という命令を実行します．モード2では割り込み認識サイクル時に，割り込み源から読み込んだ割り込みベクトルを下位8ビットとし，インタラプト・レジスタ(I)の内容を上位8ビットとして，合計16ビットのメモリ・アドレスを作り，そのアドレスのメモリから割り込みルーチンのアドレスを求め，そのアドレスへジャンプします．

〈図38〉 モード2割り込みのメカニズム

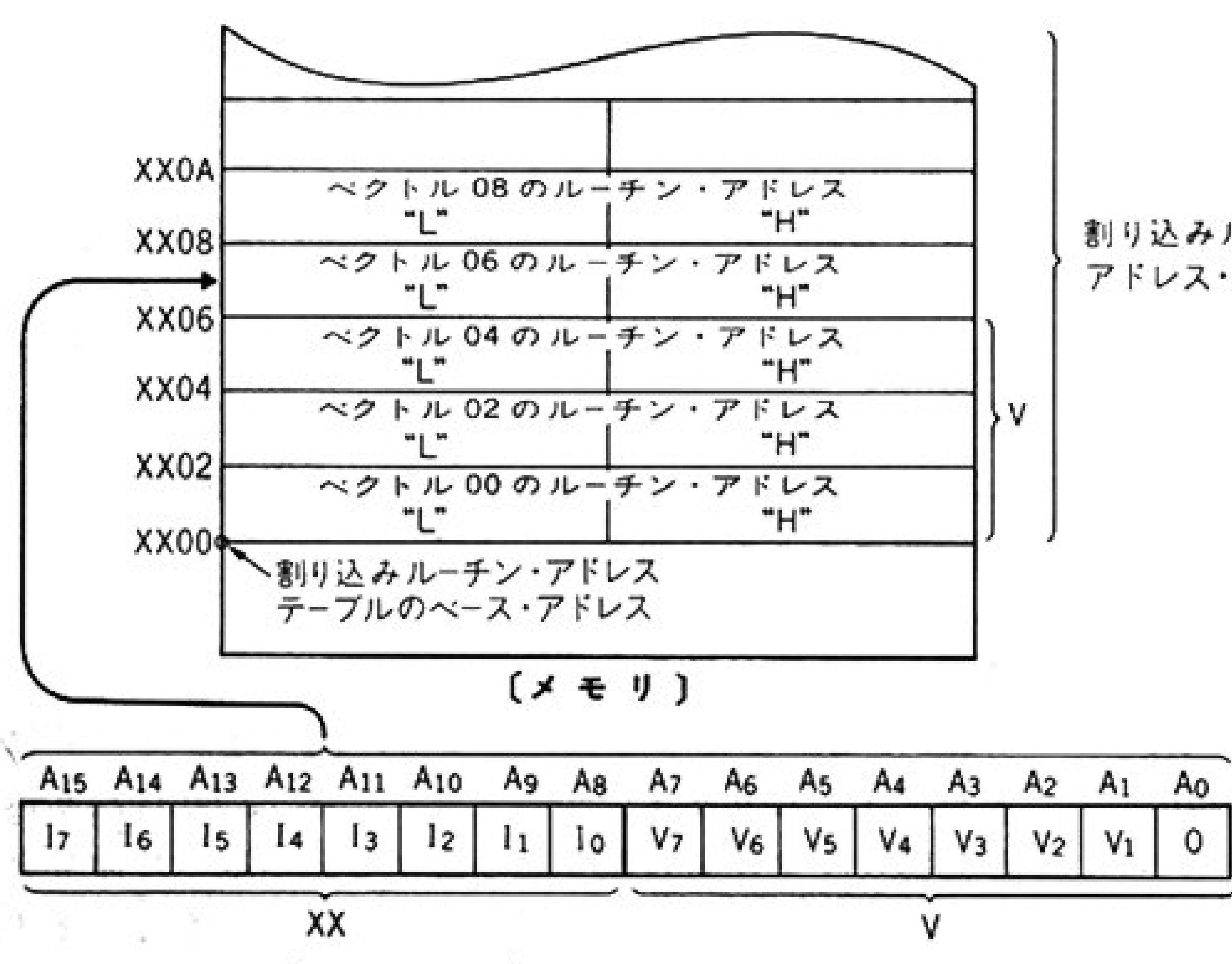
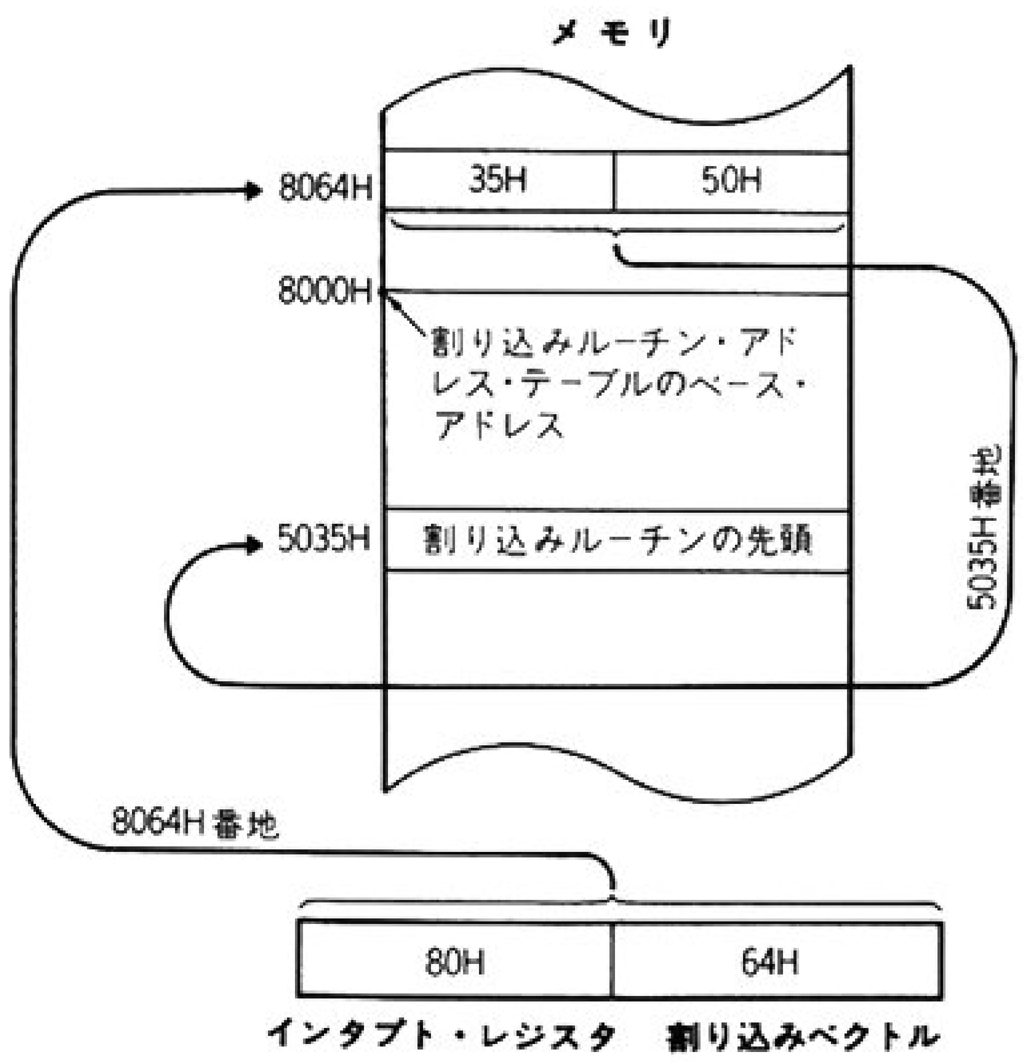


図38にモード2割り込みのメカニズムを示します．割り込みルーチン・アドレス・テーブルというのは，各割り込みベクトルの値に対応した割り込みルーチンのアドレスをセットしたテーブルであり，1ベクトルについて2バイトを占め，最高128個(256バイト)のアドレスをセットすることができます．

この割り込みルーチン・アドレス・テーブルの先頭番地は，割り込みベクトル00Hに対応し，256バイト境界(XX00H番地)にする必要があります．テーブルには，割り込みルーチンのアドレスの下位バイトを偶数番地に置き，上位バイトを

〈図 39〉 モード2割り込みの例



奇数番地にセットします．

このテーブルのアドレスの上位8ビット(XX00HのXX)は，インタラプト・レジスタ(I)の値により決まります．このインタラプト・レジスタには，LD I，Aという命令により値をセットします．

図39にしたがって，割り込み源がCPUに送った割り込みベクトルから，割り込みルーチンへジャンプする過程を説明します．この例では，インタラプト・レジスタ(I)の値は80Hです．したがって，割り込みルーチンのアドレス・テーブルは8000H番地から始まります．

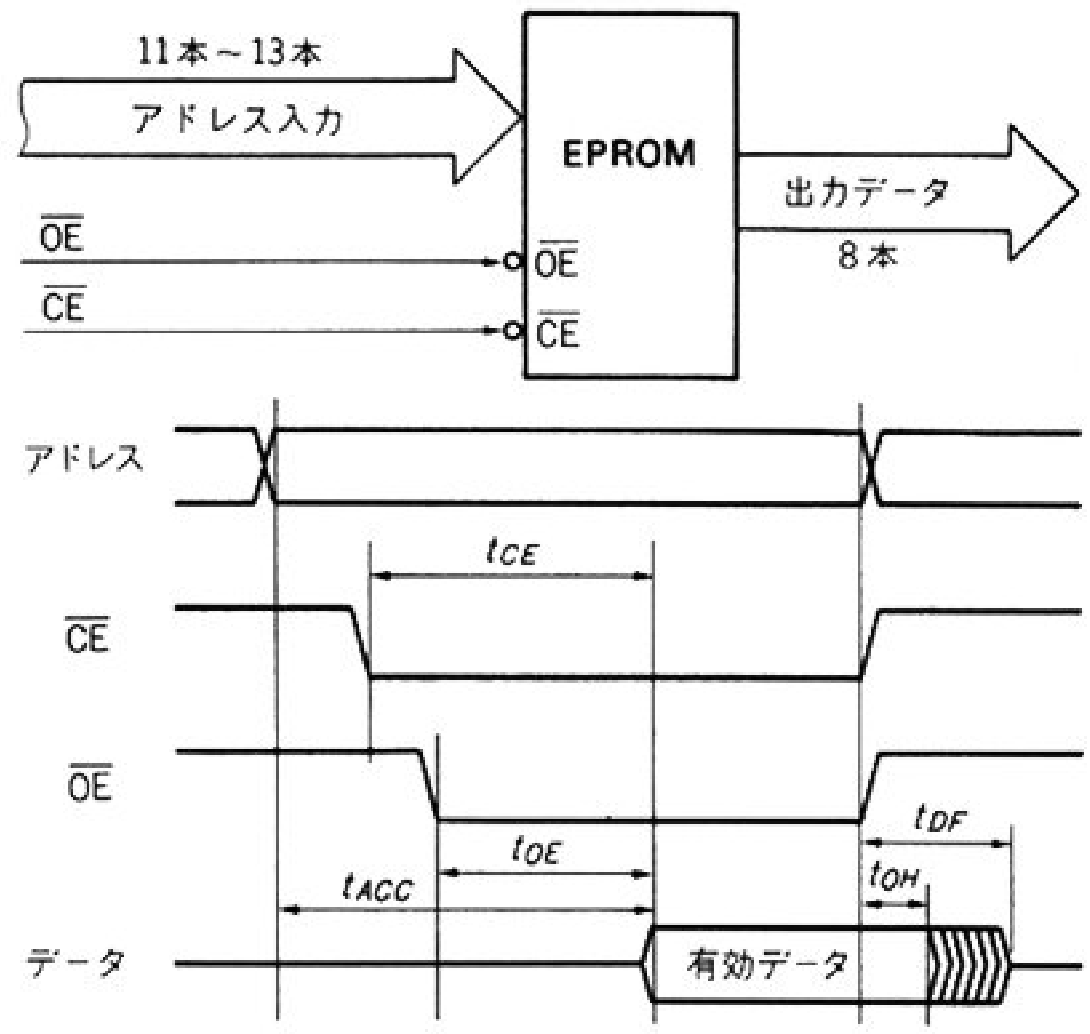
モード2割り込みが発生し，割り込み源がベクトル64HをCPUに送ったと仮定します．CPUはこのベクトル64Hとインタラプト・レジスタ(I)の内容80Hとにより，メモリの8064H番地をアクセスし，ベクトル64Hに対応する割り込みルーチンのアドレスの下位バイト(35H)を得，次に8065H番地をアクセスし，割り込みルーチンのアドレスの上位バイト(50H)を得ます．

これにより，割り込みルーチンのアドレスが5035Hであることがわかったので，CPUはベクトル64Hに対応する割り込みルーチンのアドレス，5034H番地へジャンプします．

モード2割り込みでは，割り込み源がCPUに送る割り込みベクトルのビット0は常にゼロでなければなりません．したがって，ベクトルの識別は7ビットにより行われますので，最高128種類の割り込みベクトルを持つことができます．Prog❷にモード2割り込みをセットアップするプログラムを示します．

モード2割り込みでは，最高256バイトのアドレス・テーブルを確保する必要がありますが，絶対に使わないベクトルの値に対応するアドレス・テーブルのメモリは何に使ってもかまいません．

〈図 40〉 一般的なEPROMの特性



t_{ACC}	アドレス確定後，出力データが有効になりうる時間
t_{CE}	$\overline{CE}$が"L"になってから，出力データが有効になるまでの時間
t_{OE}	$\overline{OE}$が"L"になってから，出力データが有効になるまでの時間
t_{OH}	アドレスが変化した後の出力データのホールド・タイム
t_{DF}	$\overline{OE}$が"H"になった後，出力データ・ラインがハイ・インピーダンスになるまでの時間

例としてベクトルF0H～FEHだけを使うような場合，XX00H～XXEFH(XXはインタラプト・レジスタの値)番地は割り込みとは無関係になるので，命令を置こうとデータ領域として使おうがいっこうにかまいません．

Z80 CPUとのインターフェース

これまで，“メモリからデータを読む”とか，“I/Oにデータを書き込む”というような言葉を何のためらいもなく使ってきました．しかしながら，Z80がメモリやI/Oとデータのやりとりをするためには，メモリおよびI/Oというハードウェアが必要であり，このハードウェアはZ80のタイミング，および電気的特性を守っている必要があります．

通常のメモリ・チップやI/Oチップは，そのままではCPUに接続することができず，多少の論理(ロジック)回路を介して接続します．ここではZ80 CPUとメモリやI/Oを接続するための方法について説明します．この接続を行うための論理的条件，電気的条件，それらを満足させるのに必要な論理回路等を総称して“インターフェース”といいます．

まず，はじめにZ80とEPROM，およびスタティック

〈Prog❷〉 モード２割り込みのセットアップ

```
                         ;********************************************
                         ;****    SETTING UP FOR MODE 2 INTERRUPT        ****
                         ;********************************************
0000     31 0000         BEGIN:  LD      SP,STACK        ;SET STACK POINTER
0003     21 F800                 LD      HL,TABLE        ;LOAD INTERRUPT ROUTINE
                                                         ;ADDRESS TABLE BASE
0006     7C                      LD      A,H             ;
0007     ED 47                   LD      I,A             ;SET INTERRUPT REGISTER
0009     ED 5E                   IM      2               ;SET MODE 2 INTERRUPT
000B     FB                      EI                      ;ENABLE INTERRUPT
                         ;
                         ;
                                 ORG     0185H
                         ;       INTERRUPT ROUTINE (VECTOR=00)
0185                     INT00:
                         ;
                                 ORG     0800H
                         ;       INTERRUPT ROUTINE (VECTOR=01)
0800                     INT01:
                         ;
                                 ORG     0A80H
                         ;       INTERRUPT ROUTINE (VECTOR=02)
0A80                     INT02:
                         ;
                                 ORG     0F00H
                         ;       INTERRUPT ROUTINE (VECTOR=03)
0F00                     INT03:
                         ;
                         ;
                         ;
                                 ORG     2100H
                         ;       INTERRUPT ROUTINE (VECTOR=F8)
2100                     INTF8:
                         ;
                                 ORG     4678H
                         ;       INTERRUPT ROUTINE (VECTOR=FA)
4678                     INTFA:
                         ;
                                 ORG     7800H
                         ;       INTERRUPT ROUTINE (VECTOR=FC)
7800                     INTFC:
                         ;
                                 ORG     0CF87H
                         ;       INTERRUPT ROUTINE (VECTOR=FE)
CF87                     INTFE:
                         ;
                                 ORG     0F800H
                         ;       INTERRUPT ROUTINE ADDRESS TABLE
F800     0185            TABLE:  DW      INT00           ;VECTOR=00
F802     0800                    DW      INT01           ;VECTOR=01
F804     0A80                    DW      INT02           ;VECTOR=02
F806     0F00                    DW      INT03           ;VECTOR=03
                         ;
                         ;
                                 ORG     TABLE+0F8H
F8F8     2100                    DW      INTF8           ;VECTOR=F8
F8FA     4678                    DW      INTFA           ;VECTOR=FA
F8FC     7800                    DW      INTFC           ;VECTOR=FC
F8FE     CF87                    DW      INTFE           ;VECTOR=FE
                         ;
0000                     STACK   EQU     0000H           ;STACK FRAME
                         ;
                                 END
```

RAMとのインターフェースについて説明します．ダイナミックRAMとのインターフェースについては，対象が比較的大きいので，稿を分けて説明します．

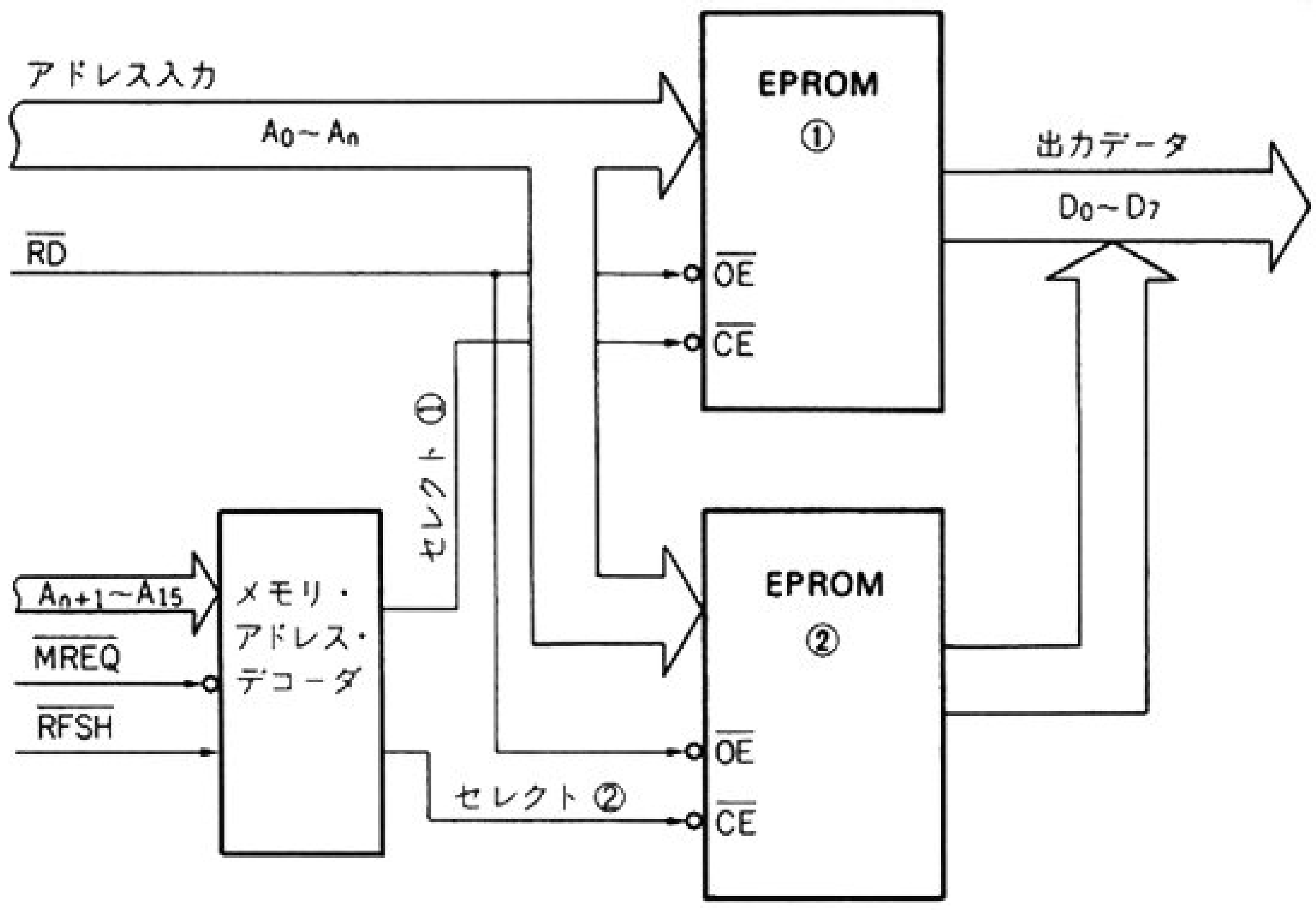
EPROMとのインターフェース

ここで取りあげるEPROMは，2716(2Kバイト)，2732(4Kバイト)，および2764(8Kバイト)の3種類です．図40に一般的なEPROMの特性を示します．EPROMはアドレス入力(11本～13本)と出力データ(8本)，それに制御信号，$\overline{OE}$(Output Enable)と$\overline{CE}$(Chip Enable)を持っています．

EPROMは$\overline{CE}$信号が"L"になると，内部の制御回路がイネーブルされ，アドレス入力をデコードすることにより，2K，4Kまたは8Kバイトあるメモリ・セルの中の1バイトを選択し，データを出力バッファにセットします．出力バッファにセットされたデータは，$\overline{OE}$信号が"L"になるとデータ・ライン上に出力されます．

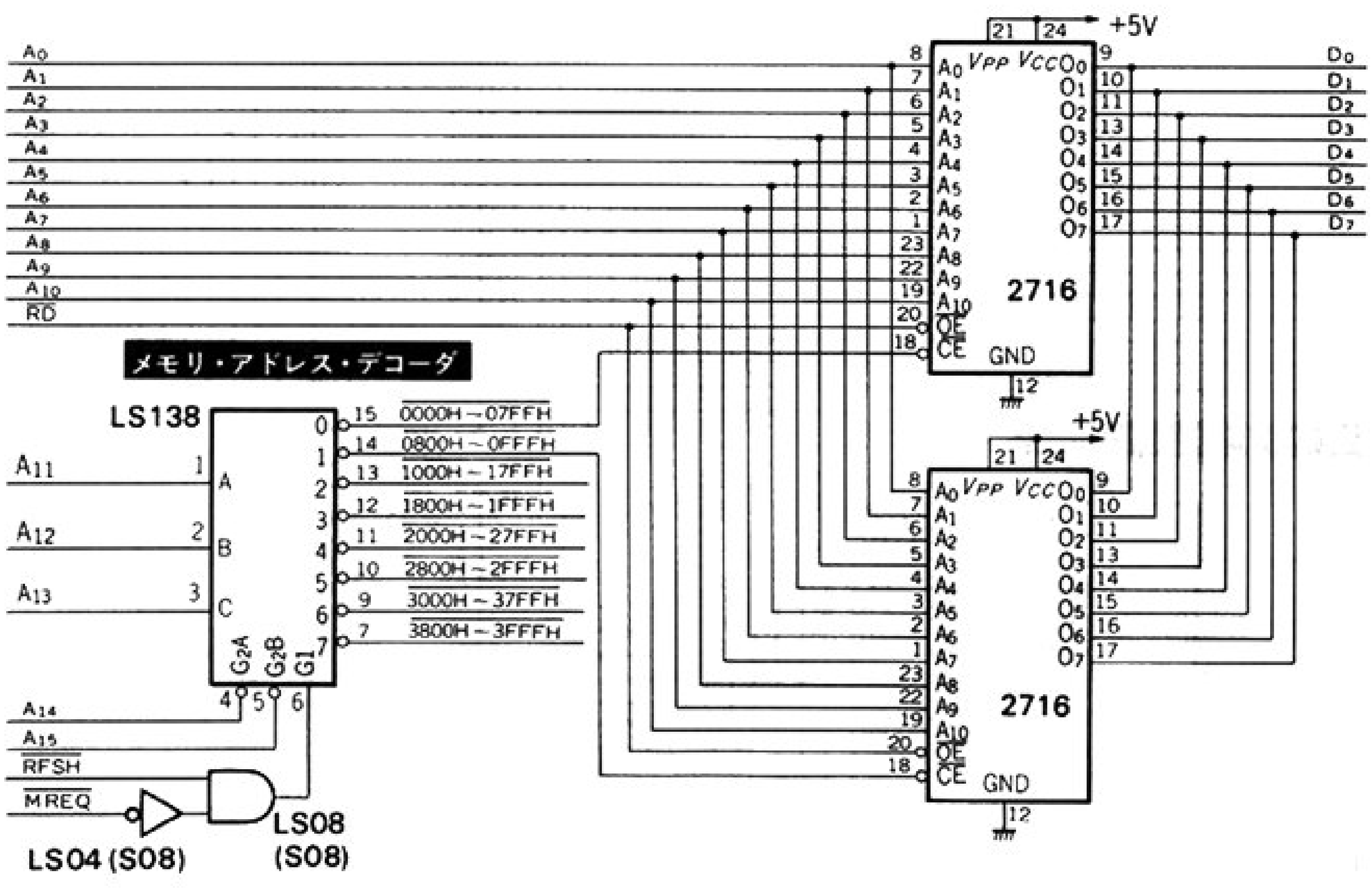
図41にZ80とEPROMとの一般的なインターフェース例を示します．EPROMのアドレス入力には，その容量に応じた数のアドレス・ラインを接続します．2Kバイトの容量であればA_0～A_{10}を，4Kバイトの場合にはA_0～A_{11}，8Kバイトの場合にはA_0～A_{12}を接続します．$\overline{OE}$入力には$\overline{RD}$信号をそのまま接続します．出力データ・ラインは，そのままZ80のデータ・バスに接続すればよく，EPROMが複数個ある場合には，ワイヤードOR接続をすることができます．

〈図41〉 Z80とEPROMとの一般的なインターフェース



最後に残ったのは各EPROMに対する$\overline{CE}$信号です．いくつかあるEPROMの中の特定のものを選択するには，選択すべきEPROMの$\overline{CE}$だけを"L"にする必要があります．この仕事を行うのが，メモリ・アドレス・デコーダです．このデコーダは，直接EPROMに接続されたアドレス以外のアドレスをデコードし，どのEPROMが選択されているのかの判定を行い，選択されているEPROMに対してのみ，$\overline{CE}$信号を出力します．

メモリ・アドレス・デコーダは，アドレスの判定を行うだけでなく，Z80が本当にメモリをアクセスしているという識別も行う必要があります．このため，$\overline{MREQ}$信号および$\overline{RFSH}$信号が，同デコーダに入力さ



〈図42〉 EPROM 2716とのインターフェース

れています．Z80はメモリからデータを読むときには，必ず$\overline{MREQ}$を“L”にします．したがって，アドレスをチェックするのと同時に$\overline{MREQ}$が“L”になっていることを確める必要があるため，$\overline{MREQ}$信号も入力されています．

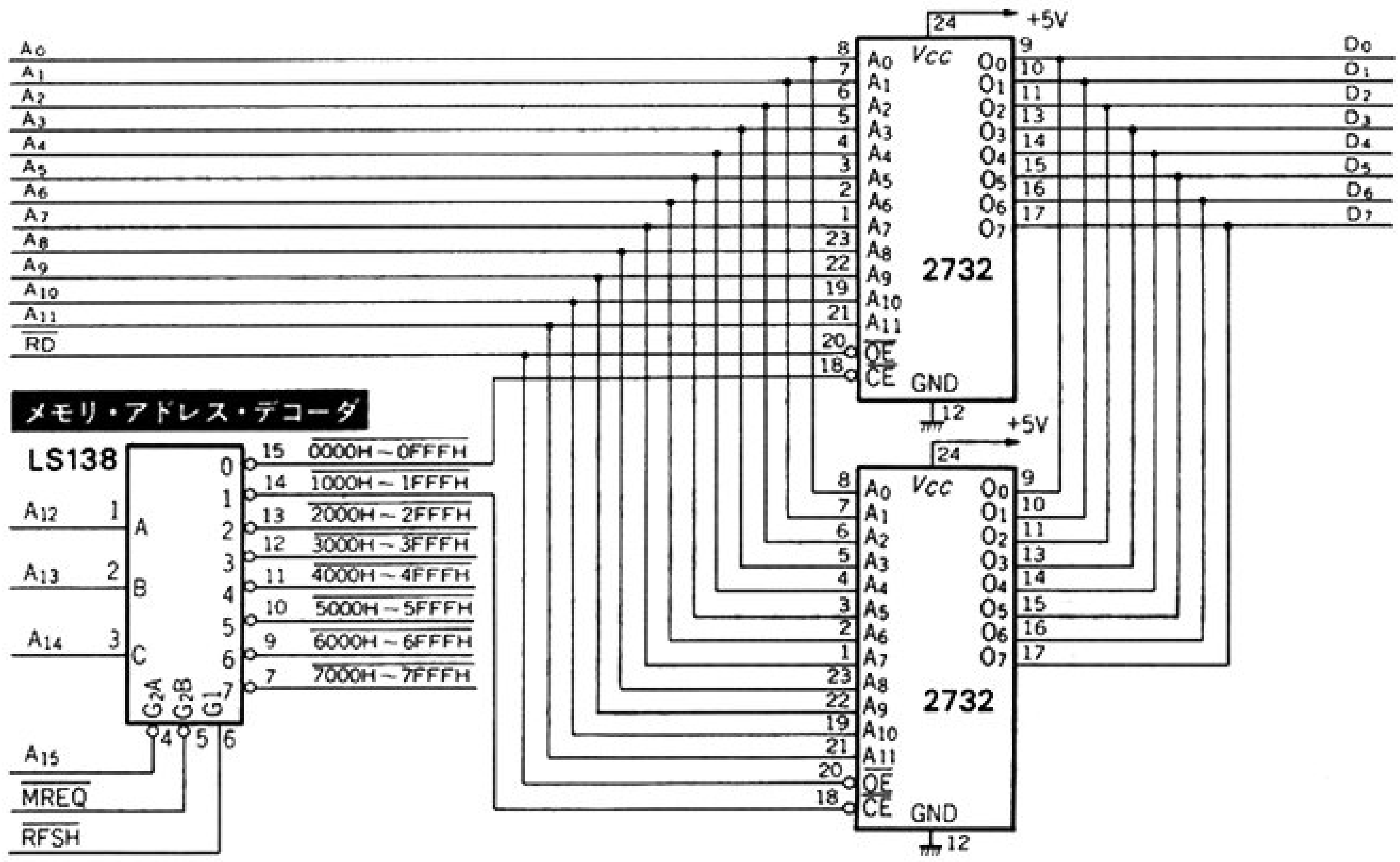
またZ80はリフレッシュ・サイクル時にも$\overline{MREQ}$を“L”にしますが，このときの$\overline{MREQ}$はメモリに対するリード動作とは関係がありませんので，$\overline{RFSH}$信号が“L”のときには，どのEPROMに対しても，$\overline{CE}$信号を出さないようにするために，$\overline{RFSH}$信号もこのデコーダに入力されています．

図42〜図44に2716，2732，2764とのインターフェース例を示します．図42の2716とのインターフェースでは，2716が2個使われており，それぞれのアドレスは0000H〜07FFHおよび0800H〜0FFFHです．メモリ・アドレス・デコーダとして，3-8デコーダのLS138を用い，これはA_{15}とA_{14}が“L”でかつ$\overline{MREQ}$信号が“L”で$\overline{RFSH}$信号が“H”のときにイネーブルされます．

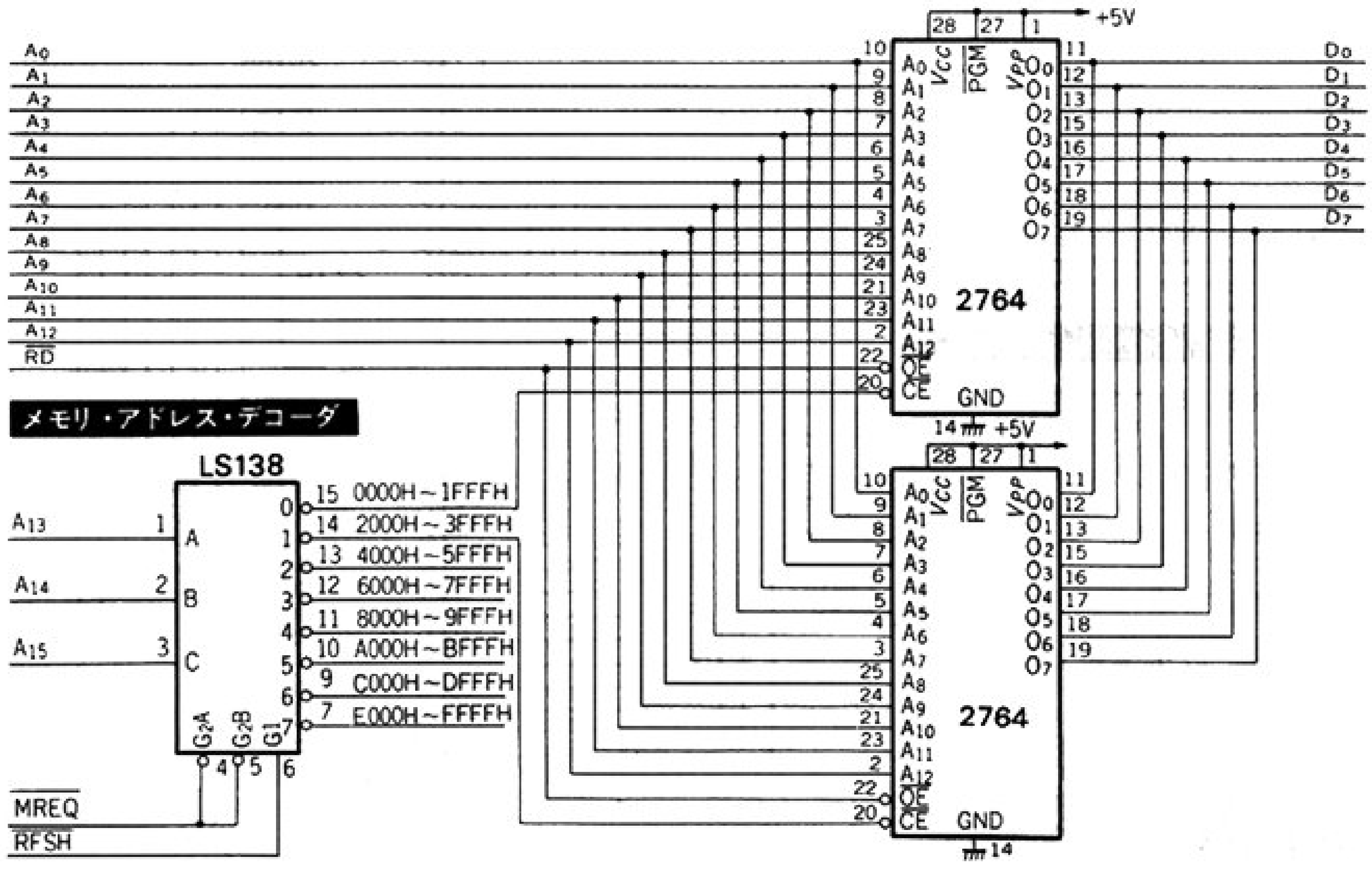
このアドレス・デコーダは，A_{11}〜A_{13}をデコードし，アドレス範囲，0000H〜3FFFH(16K)を2Kバイトごとに八つにデコードします．そして出力の下二つが2716に接続されています．2716を3個以上使用するときには，LS138の残りの出力をメモリ・アドレスに応じて，2716の$\overline{CE}$に接続すれば，図42の回路により最高8個の2716とインターフェースできます．

図43の2732とのインターフェースでも，アドレス・デコーダとしてLS138が使われています．このデコー

〈図43〉
EPROM 2732とのインターフェース



〈図44〉
EPROM 2764とのインターフェース



ダは，A_{15}と$\overline{\text{MREQ}}$信号が"L"で$\overline{\text{RFSH}}$が"H"のときにイネーブルされ，A_{12}～A_{14}をデコードして，アドレス範囲0000H～7FFFH(32K)を4Kバイトごとの八つに分けます．

図44は2764とのインターフェースです．2764は1個当たり8Kバイトなので，アドレス・デコーダはA_{13}～A_{15}をデコードし，0000H～FFFFH(64K)のアドレス範囲を8分割しています．デコーダのLS138は$\overline{\text{MREQ}}$信号が"L"で，かつ$\overline{\text{RFSH}}$信号が"H"のときにイネーブルされます．

■どれくらいのアクセス・タイムが必要か?

前項ではEPROMとZ80のインターフェースについて説明をしましたが，その説明の中には時間の概念がまったく含まれていませんでした．ここではZ80のタイミングに基づいて，EPROMをZ80に接ぐには，どれくらいのアクセス・タイムを持つ，EPROMを使えばよいかを考えてみます．

メモリを使う際，常にこのことを考えておかないと，遅すぎて使いものにならなかったり，速すぎるものを使用してお金をムダにしたりするハメに陥りかねません．

図45にZ80のM1サイクル，およびメモリ・リード・サイクルにおけるEPROMの動作タイミングを示します．このタイミング・チャートには，Z80固有の時間的定数と，EPROMおよびその周辺回路固有の時間的定数が含まれています．両者を区別するため，EPROMおよびその周辺回路に固有の時間には＊(アステリスク)を付けてあります．

この図からわかるように，必要なEPROMのアクセス・タイムは，

・M1サイクルでは，

$$T_A = 1.5\,\phi - t_{DL\overline{\phi}(MR)} - t_{DEC}{}^* - t_{S\phi(D)}$$

・メモリ・リード・サイクルでは，

$$T_B = 2.0\,\phi - t_{DL\overline{\phi}(MR)} - t_{DEC}{}^* - t_{S\phi(D)}$$

となります．

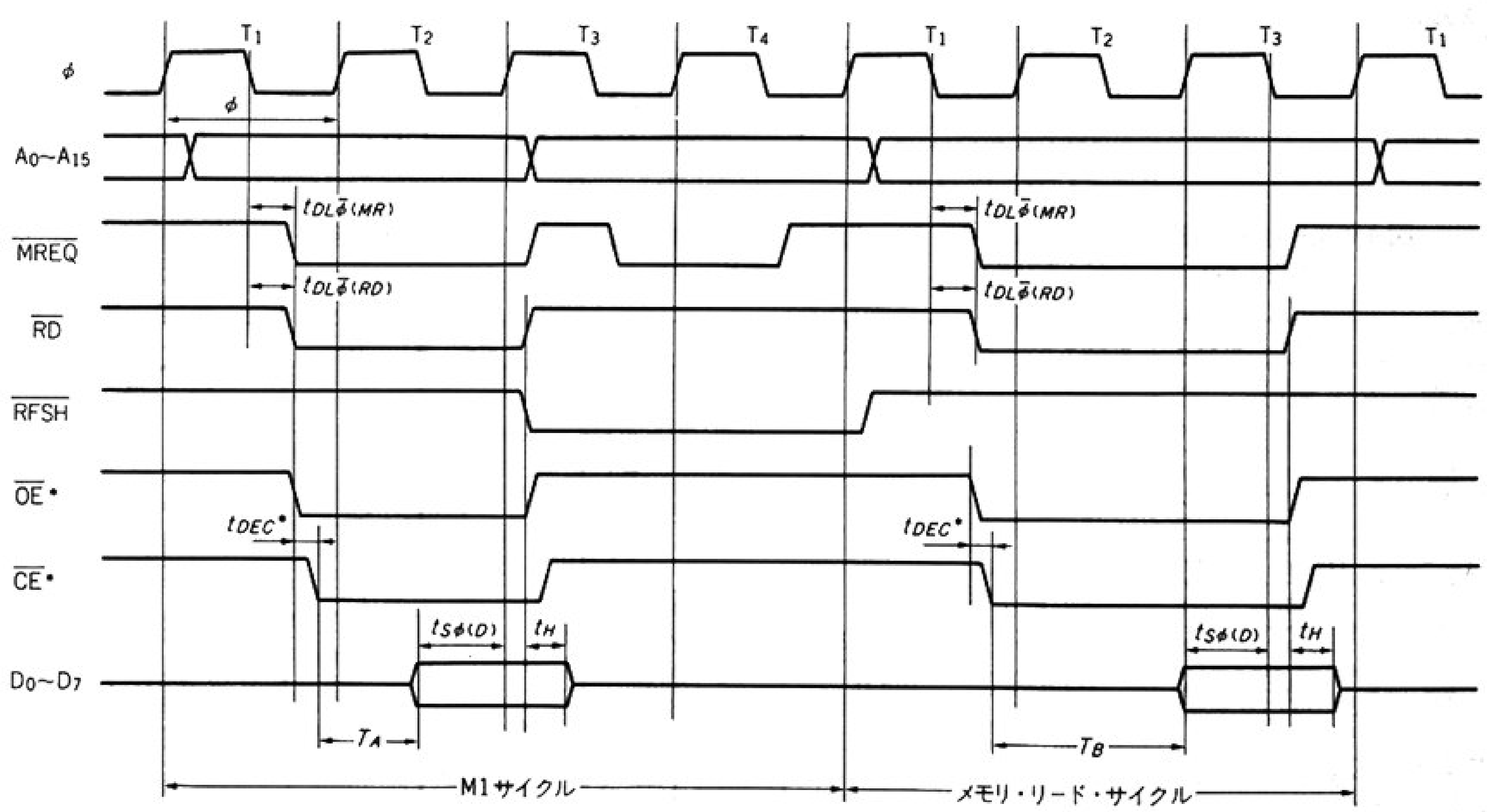
ただしϕはCPUのクロック周期であり，$t_{DEC}{}^*$はアドレス・デコーダの遅延時間です．

この二つの式からわかるように，T_AとT_Bの間には，常に$T_A \leqq T_B - 0.5\,\phi$の関係が成り立ちますので，EPROMのアクセス・タイムを考える場合，M1サイクルにおける値が最悪ケースとなります．

したがって，M1サイクルで間に合うようなアクセス・タイムを持ったEPROMを使っていれば，メモリ・リード・サイクルにおいても十分であるということができます．

T_AおよびT_Bの式に実際の値を代入し，Z80(2.5 MHz)，Z80A(4 MHz)，Z80B(6 MHz)を動作させる

〈図45〉Z80とEPROMのタイミング



T_A：$\overline{\text{CE}}$が出力されてから，データが確定する必要のある時間(M1サイクル)

$T_A = 1.5\phi - t_{DL\overline{\phi}(MR)} - t_{DEC}{}^* - t_{S\phi(D)}$

T_B：$\overline{\text{CE}}$が出力されてから，データが確定する必要のある時間(メモリ・リード・サイクル)

$T_B = 2.0\phi - t_{DL\overline{\phi}(MR)} - t_{DEC}{}^* - t_{S\phi(D)}$

ϕ：クロック周期　　$t_{DEC}{}^*$：アドレス・デコーダの遅延時間

のに必要な，EPROMのアクセス・タイムを図46に示します．この図からわかるようにZ 80，Z 80 A，Z 80 Bを用いる場合に必要なEPROMのアクセス・タイムは，それぞれ375 ns，217 ns，112.5 nsとなります．

では現在入手可能なEPROMのアクセス・タイムはどれくらいなのでしょうか．インテル社のデータ・ブックから，ここで取りあげたEPROMの時間的定数(遅延時間)を拾い出し，図47に示します．

図42〜図44の回路では，$\overline{OE}$には$\overline{RD}$信号を直接接続し，アドレス・デコーダの出力を$\overline{CE}$に接続してありますので，EPROMのアクセス・タイムとして，t_{CE}をとる必要があります．t_{CE}で一番小さな値を持っているのは，2732 A-2(200 ns)と2764-2(200 ns)ですので，図42〜図44の回路を用いる限り，Z 80 Bに対応できるEPROMは今のところないということができます．

データのホールド・タイムについて一言しますと，Z 80のホールド・タイムは0であり，EPROMの場合$\overline{CE}$または$\overline{OE}$が“H”になってからのデータのホールド・タイムは100 nsくらいありますので，設計に際して何の考慮も入れる必要はありません．

ここで述べたEPROMのアクセス・タイムに関する説明は，後に述べるスタティックRAMのアクセス・タイムについても適用できます．

〈図46〉必要なEPROMのアクセス・タイム
〔図45のT_AとT_Bの値(max)〕

時間 / タイプ	ϕ	$t_{DL\overline{\phi}(MR)}$	t_{DEC}*	$t_{S\phi(D)}$	t_A	t_B
Z 80	400	100	73	50	375	575
Z 80 A	250	85	38	35	217	342
Z 80 B	167	70	38	30	112.5	196

〔単位はns〕

LS 04	LS 08	LS138
15	20	38

■アクセス・タイムを稼ぐ方法

前項の説明からおわかりのように，Z 80 AおよびZ 80 BをCPUとし，それらを最高速度で動かすと，相当速いEPROMが必要となります．メモリの速度がCPUの速度に追いつかないときには，ウェイト・ステートを用いるのが常とう手段ですが，ここでは別の方法を試みます．

前項では，EPROMの$\overline{CE}$信号を$\overline{MREQ}$信号が“L”になった時点に出力し，$\overline{OE}$信号として$\overline{RD}$信号を用いていました．したがって，EPROMは$\overline{MREQ}$信号が“L”になるより，ずいぶん前にアドレスが確定しているにもかかわらず，$\overline{MREQ}$信号が“L”になるまで，メモリ・セルのアクセスを開始することができません．

ここでは，EPROMの$\overline{CE}$を常に“L”にし，$\overline{MREQ}$信号により$\overline{OE}$を作ってみることにします．こうすると，Z 80のアドレス・バスの状態が確定した時点から，EPROMのアクセスが開始されます．

図46からわかるように，EFROMの特性は$\overline{CE}$が“L”になってからのアクセス・タイム(t_{CE})と，アドレスが確定してからのアクセス・タイム(t_{ACC}：ただし$\overline{CE}$=“L”)の値は同じです．

よってZ 80は，$\overline{MREQ}$信号が“L”になるよりも前にアドレスを確定しますので，EPROMの$\overline{CE}$を常に“L”にし，$\overline{MREQ}$信号により$\overline{OE}$を作ることにより，アクセス・タイムを稼ぐことができます．$\overline{OE}$が“L”になって

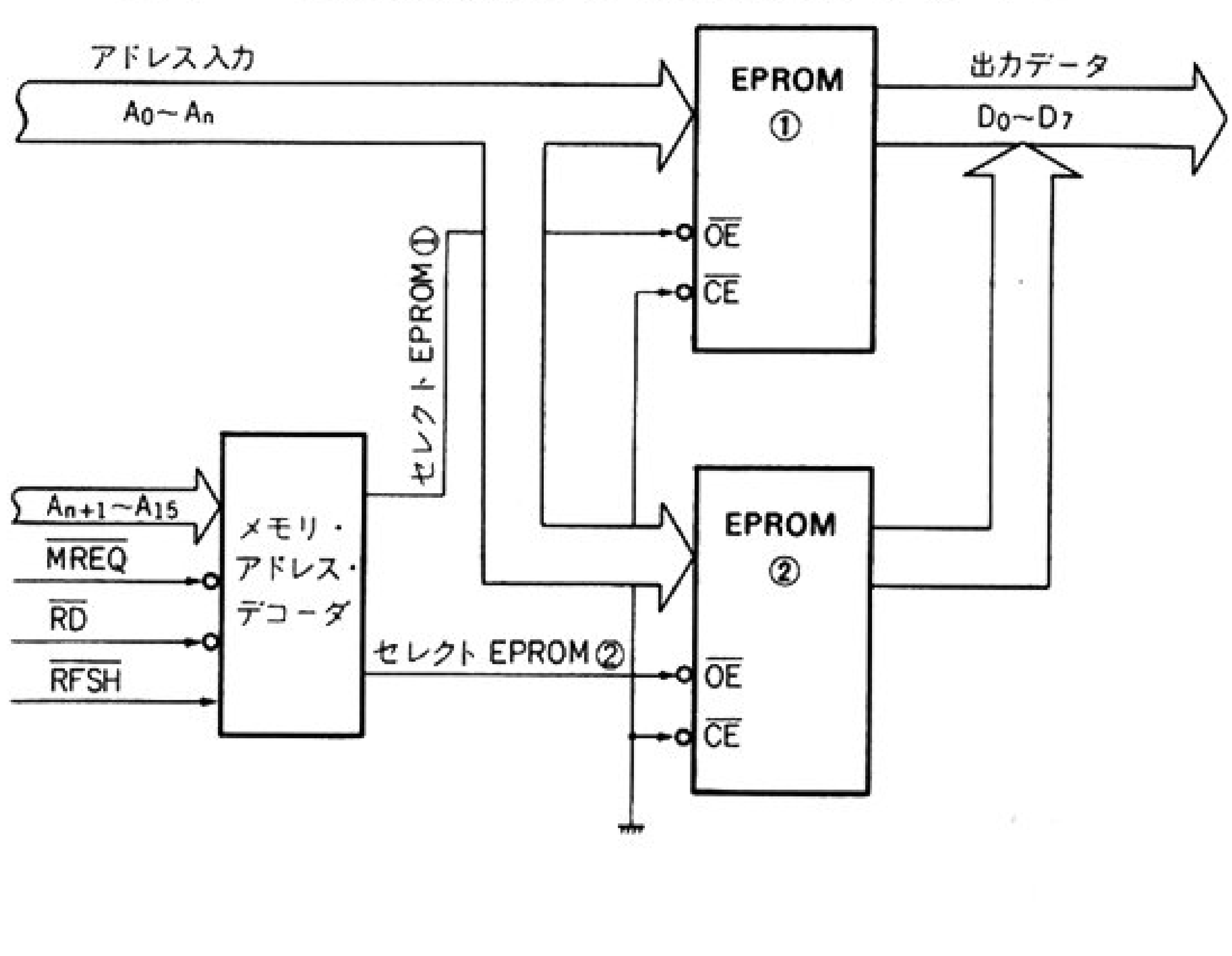
〈図47〉EPROMの遅延時間
(単位：ns)

時間 / タイプ	2716	2716-1	2716-2
t_{ACC}	450	350	390
t_{CE}	450	350	390
t_{OE}	120	120	120

時間 / タイプ	2732A	2732A-2	7232A-3
t_{ACC}	250	200	300
t_{CE}	250	200	300
t_{OE}	100	70	150

時間 / タイプ	2764	7264-2	2764-3
t_{ACC}	250	200	300
t_{CE}	250	200	300
t_{OE}	100	70	150

〈図48〉EPROMの$\overline{CE}$を常に“L”にした場合のインターフェース



から，出力データが出るまでの時間は，EPROMのアクセス・タイム(t_{ACC}, t_{CE})に比べて，十分小さいからです．

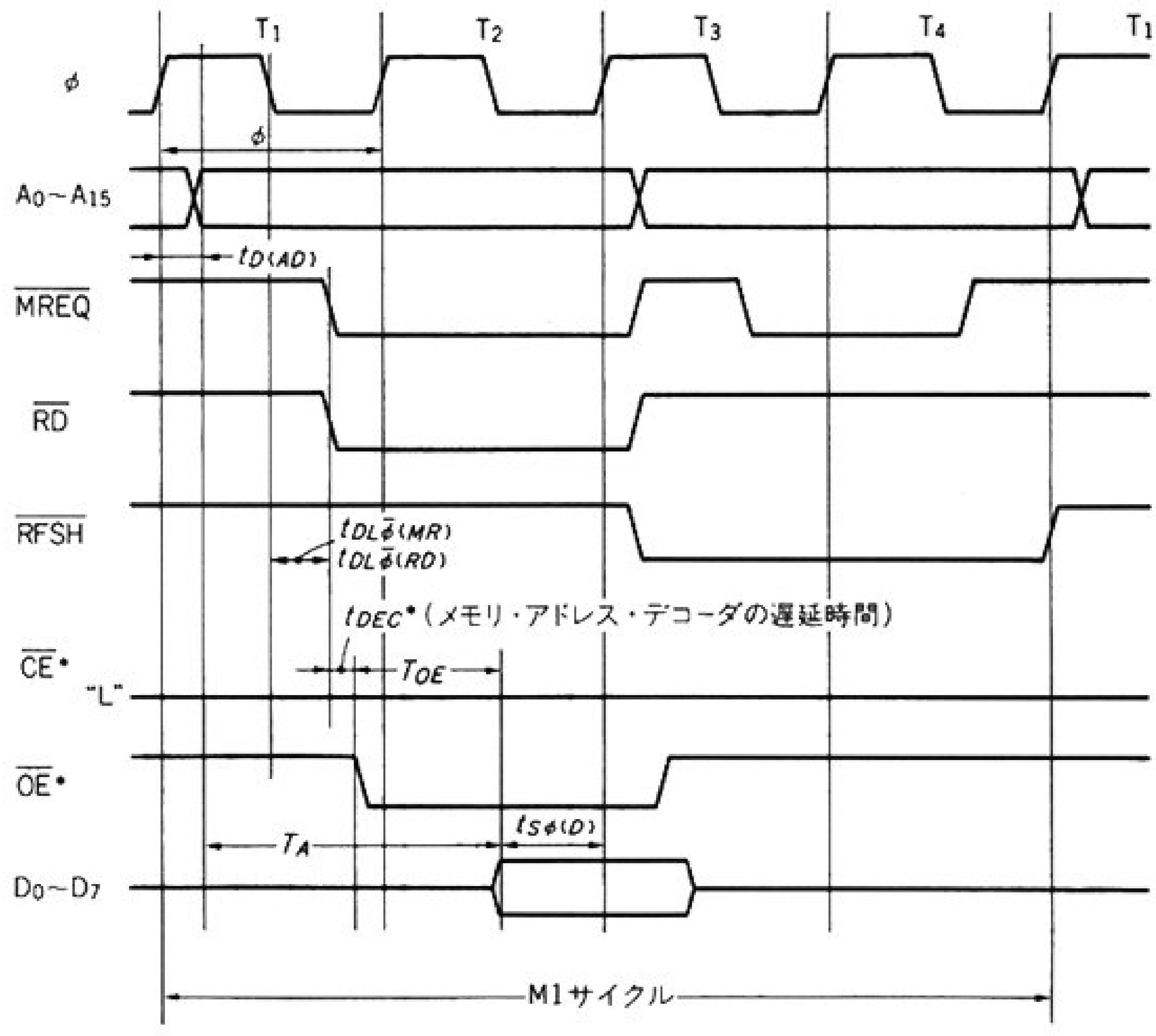
図48に EPROMの$\overline{\text{CE}}$を常に“L”にし，$\overline{\text{MREQ}}$信号により，$\overline{\text{OE}}$を作った場合のインターフェースを示します．また この場合のZ 80 とEPROMのタイミングを図49に示します．前に説明しましたが，M1サイクルにおける場合が最悪ケースですので，この図にはM1サイクルだけしか示してありません．

この図からわかるように，$\overline{\text{CE}}$を常に“L”にした場合に必要なEPROMのアクセス・タイムT_Aは，

$$T_A = 2\,\phi - t_{D(AD)} - t_{S\phi(D)}$$

となります．図50にZ 80, Z 80 A, Z 80 Bをそれぞれ最高速度で動作させた場合のT_Aの値を示します．図45のT_Aの値と比べていただければ，ここで述べた方法によりアクセス・タイムを相当稼ぐことができることが，おわかりになると思います．

〈図49〉 $\overline{\text{CE}}$を常に“L”にした場合のZ 80 とEPROMのタイミング



$T_A = 2\phi - t_{D(AD)} - t_{S\phi(D)}$, $T_{OE} = 1.5\phi - t_{DL\overline{\phi}(MR)} - t_{DEC}{}^{*} - t_{S\phi(D)}$

ここで心配になるのは，$\overline{\text{OE}}$が“L”になってから，EPROMの出力データが確定するまでの時間(図49のT_{OE})です．図49から，

$$T_{OE} = 1.5\,\phi - t_{DL\overline{\phi}(MR)} - t_{DEC}{}^{*} - t_{S\phi(D)}$$

となります．

図51にZ 80，Z 80 A，Z 80 Bをそれぞれ最高速度で動作させた場合に必要なT_{OE}の値を示します．図47からわかるように，EPROMの t_{OE} の値は 70 ns～120 ns なので，Z 80 Bを 6 MHzで動作させた場合にも，特殊なEPROMを使う必要はなく，ふつうのもので間に合うということができます．

〈図50〉 $\overline{\text{CE}}$を常に“L”にした場合に必要なEPROMのアクセス・タイムT_A(単位：ns)

タイプ＼時間	ϕ	$t_{D(AD)}$	$t_{S\phi(D)}$	t_A
Z80	400	145	50	605
Z80A	250	110	35	355
Z80B	167	90	30	214

$t_A = 2\phi - t_{D(AD)} - t_{S\phi(D)}$

〈図51〉 $\overline{\text{CE}}$を常に“L”にした場合，$\overline{\text{OE}}$が“L”になってから出力データが確定する必要のある時間T_{OE}(単位：ns)

タイプ＼時間	ϕ	$t_{DL\overline{\phi}(MR)}$	t_{DEC*}	$t_{S\phi(D)}$	t_{OE}
Z80	400	100	73	50	377
Z80A	250	85	73	35	182
Z80B	167	75	73	30	72.5

$t_{OE} = 1.5\phi - t_{DL\overline{\phi}(MR)} - t_{DEC*} - t_{S\phi(D)}$

スタティックRAMとのインターフェース

ここで取りあげるスタティックRAMは，2 K×8 ビット構成の6116です．この6116はリード/ライト可能なメモリですので，ライト・イネーブル($\overline{\text{WE}}$)端子を持ち，かつデータ・ラインは双方向性ですが，リード動作に関しては，EPROMの 2716 とまったく同じ動きをします．

図52にZ 80 とスタティックスRAM 6116 とのインターフェースのブロック図を示します．また図53に回路図を示します．前出の図42の 2716 とのインターフェース回路図と比べてみると，ピン㉑の信号が異なるのと，メモリ・アドレス・デコーダの回路がちょっと違っているだけであることがおわかりになることでしょう．メモリ・アドレス・デコーダの回路が異なるのは，RAMのアドレスを 8000 Hから取ったためです．

まず 6116 の動作から説明していきます．図54に，6116のリード・サイクルのタイミングを示します．6116 からデータを読む場合には，アドレスA_0～A_{10}をセットし，チップ・セレクト($\overline{\text{CS}}$)およびアウトプット・イネーブル($\overline{\text{OE}}$)を“L”にします．

$\overline{\text{CS}}$を“L”にすると，6116 はメモリ・アクセスを開始し，A_0～A_{10}により 2 Kバイトあるメモリ・セル中の 1

バイトを選択し，データをアウトプット・バッファにセットします．このアウトプット・バッファのデータは$\overline{OE}$を“L”にすることにより，データ・ライン(I/O_0～I/O_7)上に出力されます．リード・サイクル中は，ライト・イネーブル($\overline{WE}$)は“H”にしておく必要があります．

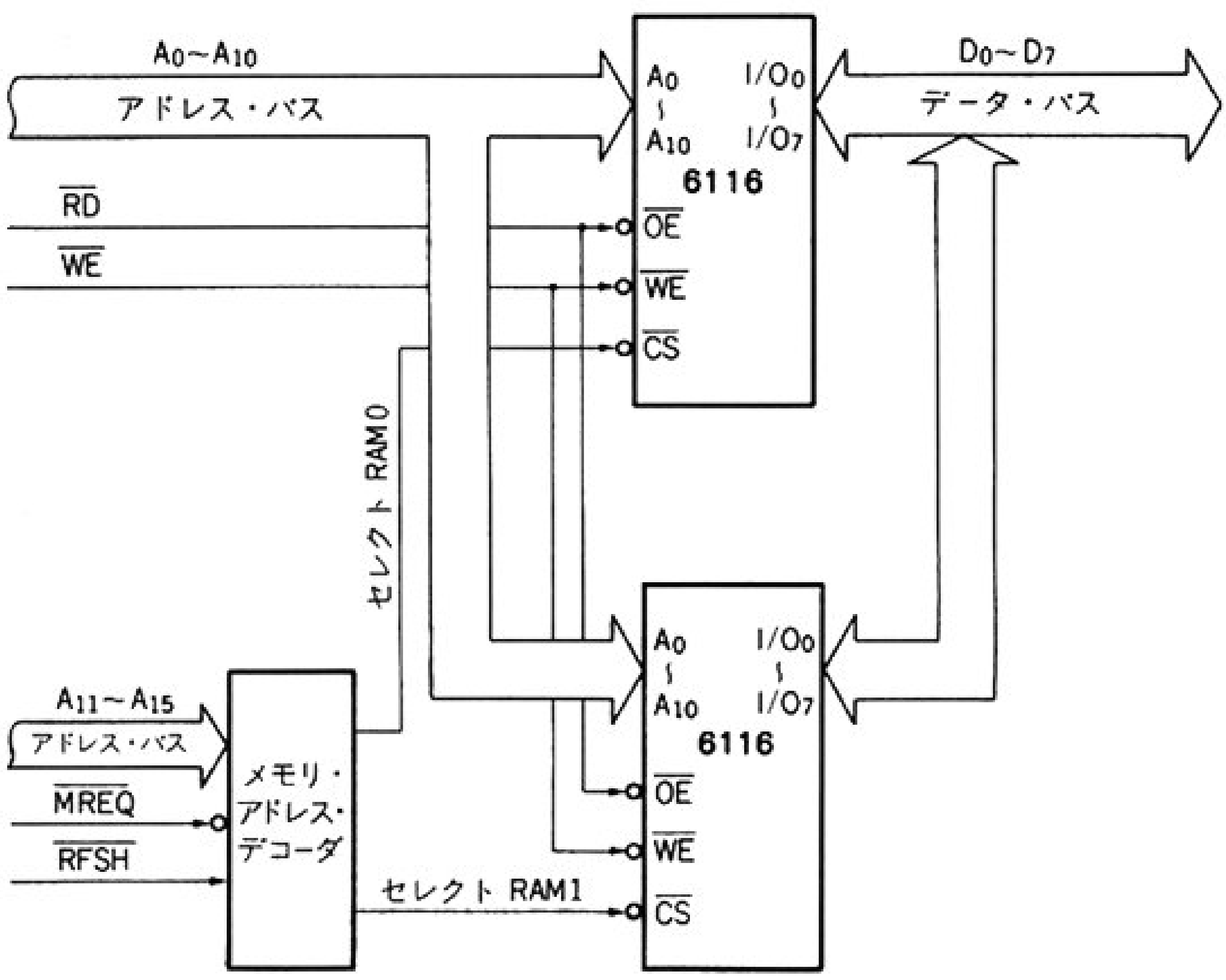
図53の回路では$\overline{MREQ}$=“L”，$\overline{RFSH}$=“H”で，かつA_{15}=“H”，A_{14}=“L”のときに，アドレス・デコーダ(LS 138)がイネーブルされ，A_{11}～A_{13}の値にしたがって，アドレス・デコーダから6116に対してチップ・セレクト信号($\overline{CS}$)が出力されます．6116の$\overline{OE}$には$\overline{RD}$信号が直接接続されています．6116の$\overline{OE}$の状態は$\overline{CS}$=“L”でないときは無効なので，このような接続が可能です．

6116のリード・サイクルにおけるアクセス・タイムを考える場合，図45のEPROMのタイミングをそのまま適用することができます．したがって図45の値をそのまま用いることができますので(LS 08とLS 00の差はあるが，ほとんど同じ)，必要なリード・アクセス・タイムは，Z 80，Z 80 A，Z 80 Bを用いる場合，それぞれ375 ns，217 ns，112.5 nsとなります．

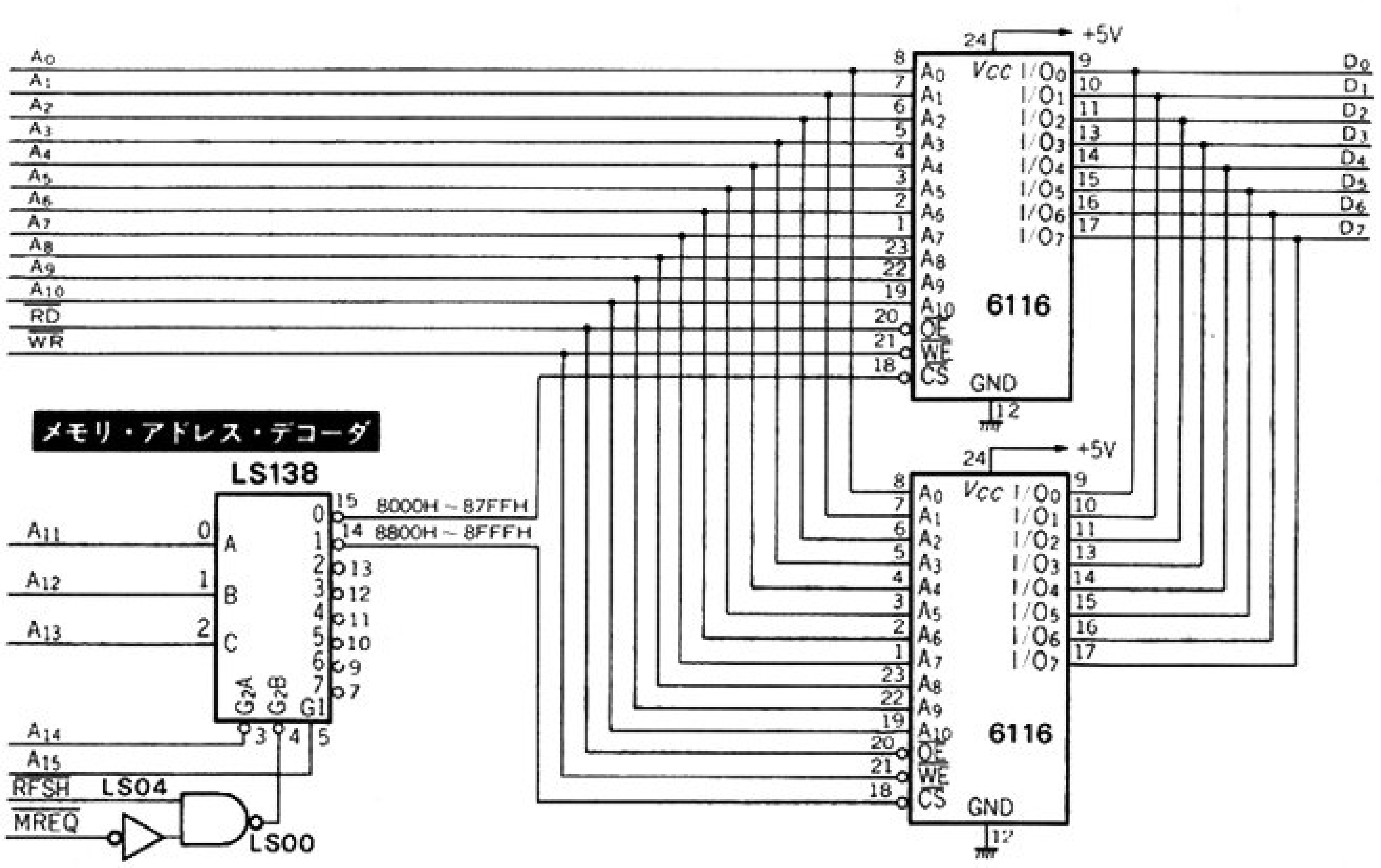
それでは6116のアクセス・タイムはどれくらいであるかというと，図54のt_{ACS}($\overline{CS}$アクセス・タイム)をごらんください．これによると，$\overline{CS}$が“L”になってからの6116のアクセス・タイムは，6116/P-2が120 ns，6116/P-3が150 ns，6116/P-4が200 nsです．

したがって，Z 80およびZ 80 Aの場合には，アクセス・タイムが200 nsの6116/P-4で十分間に合います．しかしながら，Z 80 Bの場合には必要なアクセス・タイムは112.5 nsであり，一番速い6116/P-2のアクセス・タイムは120 nsです．したがって図53の回路そのままでは，Z 80 Bと6116とのインターフェースはできないということができます．しかしながら，図53のLS 138をS 138に置き換えることにより，t_{DEC}*が27 ns減ります(遅延時間：LS 138=38 ns，S 138=11 ns)．こうすればZ 80 Bを用いる場合に必要なアク

〈図52〉Z 80とスタティックRAM 6116とのインターフェース・ブロック図

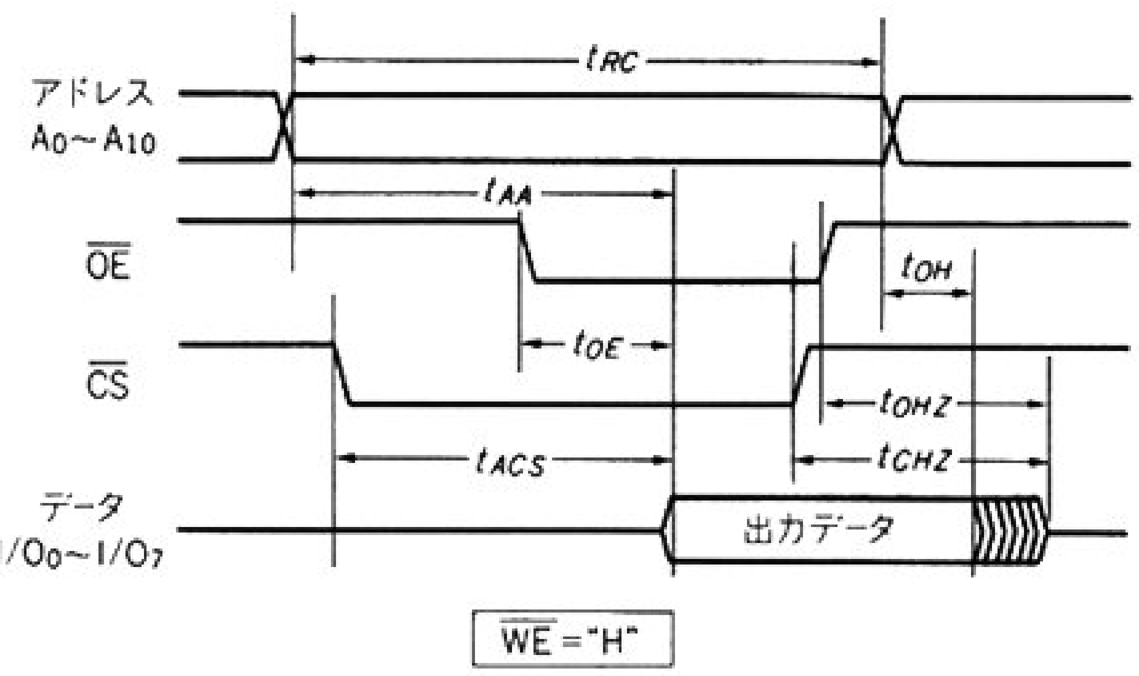


〈図53〉Z 80とスタティックRAM 6116とのインターフェースの回路図



セス・タイムは139.5nsとなり，6116/P-2が使えることになります．このようにCPUの速度が速くなると，LS-TTLではダメで，Sタイプを使わなければならないことがよくあります．

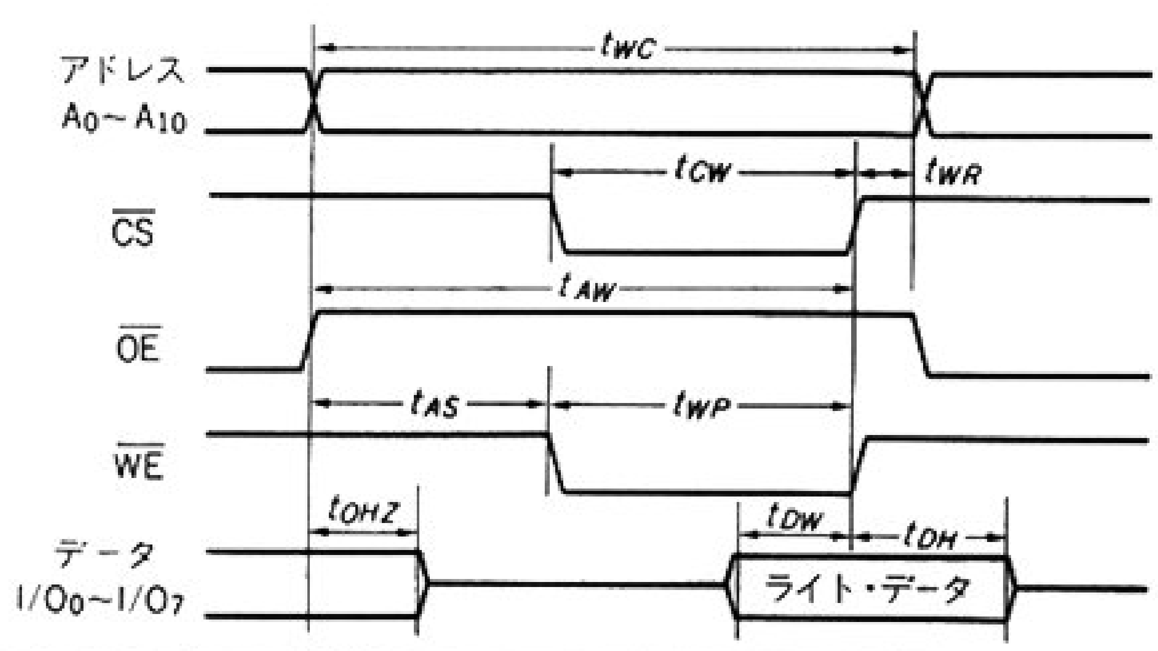
次に6116のライト・サイクルのタイミングについて説明をします．図55に6116のライト・サイクルのタイミングを示します．6116に対してデータを書き込む場合には，まずアドレス(A_0～A_{10})をセットし，$\overline{CS}$を“L”にします．ついでデータ・ライン(I/O_0～I/O_7)上に書き込むべきデータを乗せ，$\overline{WE}$を“L”にします．この際$\overline{OE}$は“H”にして，6116内部のアウトプット・バッファをディスエーブルしておく必要があります．



〈図54〉
6116のリード・サイクル・タイミング

項　目	記号	6116/P-2	6116/P-3	6116/P-4	説　明
リード・サイクル・タイム	t_{RC}	120*	150*	200*	1回のリード・サイクルに要する時間
アドレス・アクセス・タイム	t_{AA}	120	150	200	$\overline{CS}$="L"のときに，アドレスが確定してからデータがアウトプット・バッファにセットされるまでの時間
$\overline{CS}$ アクセス・タイム	t_{ACS}	120	150	200	$\overline{CS}$="L"になってから，データがアウトプット・バッファにセットされるまでの時間
$\overline{OE}$ 出力遅延時間	t_{OE}	80	100	120	$\overline{OE}$="L"になってから，アウトプット・バッファのデータがデータ・ライン上に出力されるまでの時間
出力ホールド・タイム	t_{OH}	10	15	15	アドレスが変化しても，出力データが変化しないでいる時間
$\overline{OE}$ 出力遅延時間	t_{OHZ}	40	50	60	$\overline{OE}$が"H"になってから，出力データ・ラインがハイ・インピーダンスになるまでの時間
$\overline{CS}$ 出力遅延時間	t_{CHZ}	40	50	60	$\overline{CS}$が"H"になってから，データ・ラインがハイ・インピーダンスになるまでの時間

〔単位ns，　*を付したものは最小値，その他は最大値〕



〈図55〉
6116のライト・サイクル・タイミング

項　目	記号	6116/P-2	6116/P-3	6116/P-4	説　明
ライト・サイクル・タイム	t_{WC}	120	150	200	1回のライト・サイクルに要する時間
$\overline{CS}$ パルス幅	t_{CW}	70	90	120	$\overline{CS}$のパルス幅
アドレス有効時間	t_{AW}	105	120	140	$\overline{WE}$が"H"になる前に，アドレスが確定していなければならない時間
アドレス・セットアップ・タイム	t_{AS}	20	20	20	$\overline{WE}$が"L"になる前に，アドレスが確定していなければならない時間
$\overline{WE}$ パルス幅	t_{WP}	70	90	120	$\overline{WE}$のパルス幅
アドレス・ホールド・タイム	t_{WR}	5	10	10	$\overline{CS}$または$\overline{WE}$が"H"になってから，アドレスを保持しておく必要のある時間
$\overline{OE}$ 出力遅延時間	t_{OHZ}	40*	50*	60*	$\overline{OE}$が"H"になってから，データ・ラインがハイインピーダンスになるまでの時間
データ・セットアップ・タイム	t_{DW}	35	40	60	$\overline{WE}$または$\overline{CS}$が"H"になる前に，データを確定しておく必要のある時間
データ・ホールド・タイム	t_{DH}	5	10	10	$\overline{WE}$または$\overline{CS}$が"H"になってから，データを保持しておく必要のある時間

〔単位：ns　*を付したものは最大値，その他は最小値〕

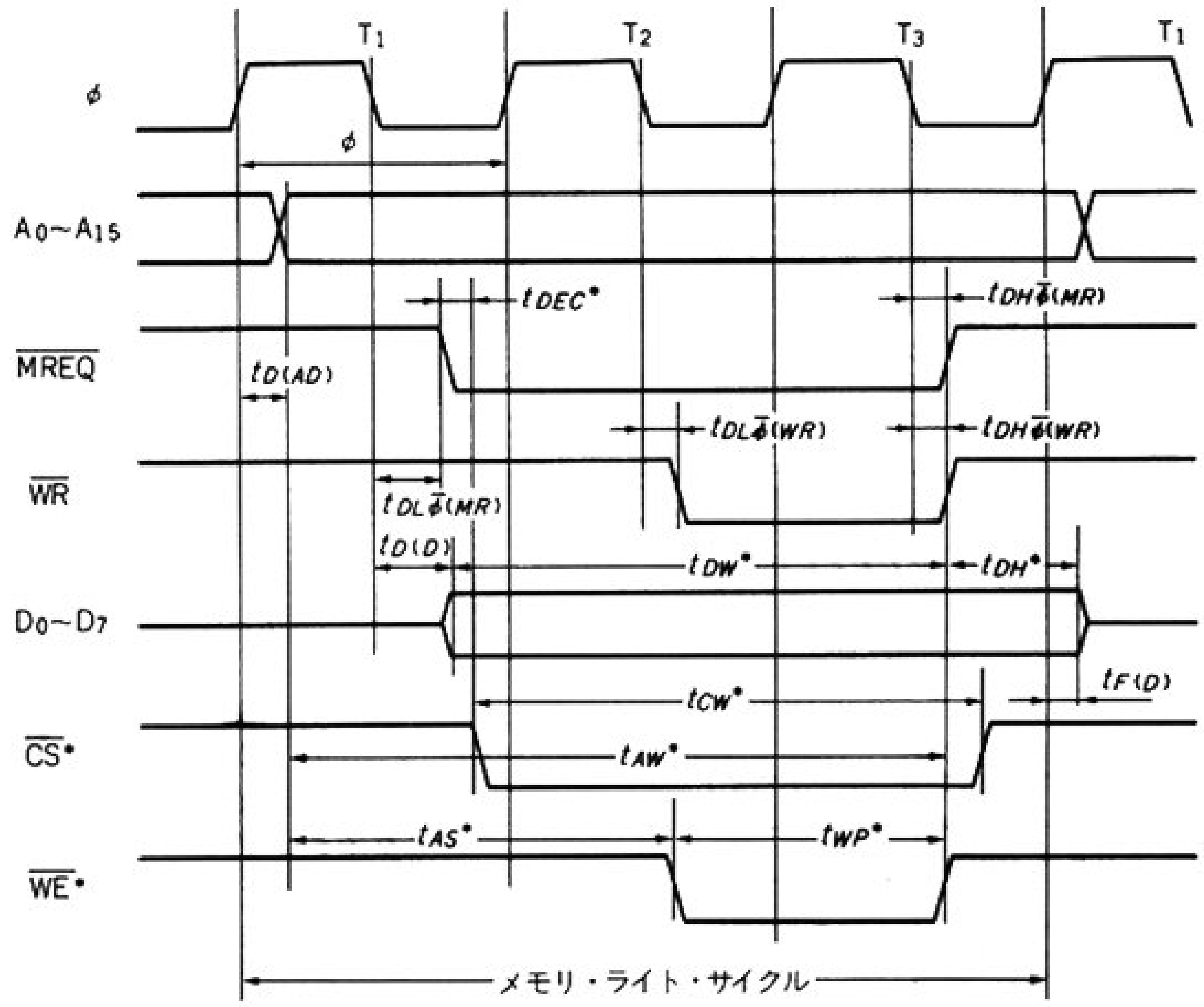
図53からメモリ・ライト・サイクルの動作を追ってみることにします．Z80のメモリ・ライト・サイクルでは，まずメモリ・アドレス(A_0〜A_{15})が出力され，次にデータ・バス(D_0〜D_7)上にデータが乗せられ，$\overline{MREQ}$信号が"L"になり，少し時間をおいて$\overline{WR}$信号が"L"になります．

メモリ・アドレス・デコーダは，$\overline{MREQ}$＝"L"，$\overline{RFSH}$＝"H"で，かつA_{15}＝"H"，A_{14}＝"L"の条件で，A_{11}〜A_{13}の値により，デコーダの出力としてのチップ・セレクト信号($\overline{CS}$)を6116に供給します．そして$\overline{WR}$が"L"になると，データ・バス上のデータが，A_0〜A_{10}により選択された6116中のメモリ・セルに書き込まれます．

図56にZ80のメモリ・ライト・サイクルにおける，Z80自体と6116，およびメモリ・アドレス・デコーダのタイミングを示します．この図中の＊印(アステリスク)を付けた時間は，Z80のメモリ・ライト・サイクルにおいて，図53の回路が動作する場合の，6116に関連した時間であり，これらの時間は図55に示した値を満足していなければならないものです．

図53の回路がZ80のメモリ・ライト・サイクルで動作した場合の，各時間値を図57に示します．この図と図55を比べるとわかるように，図57に示した値はZ80，Z80A，Z80Bのいずれとインターフェースした場合にも，6116のタイミングを満足しています．

〈図56〉Z80と6116のライト・サイクルのタイミング



t_{DEC}*：メモリ・アドレス・デコーダの遅延時間
6116のタイミングに対応する時間に，＊印がつけてある

〈図57〉6116とZ80をインターフェースした場合の各信号の時間
(メモリ・ライト・サイクル)

時　間	求める式	Z80	Z80A	Z80B
t_{AW}*	$2.5\phi - t_{D(AD)} + \underline{t_{DH\bar{\phi}(WR)}}$	855	515	327.5
t_{AS}*	$1.5\phi - t_{D(AD)} + \underline{t_{DL\bar{\phi}(WR)}}$	455	265	160.5
t_{CW}*	$2.0\phi - t_{DL\bar{\phi}(MR)} + \underline{t_{DH\bar{\phi}(MR)}}$	700	415	264
t_{WP}*	最小値が決められている	360	220	135
t_{DW}*	$2.0\phi - t_{D(D)} + \underline{t_{DH\bar{\phi}(WR)}}$	570	350	204
t_{DH}*	$0.5\phi - t_{DH\bar{\phi}(WR)} + \underline{t_{F(D)}}$	100	45	13.5

(単位：ns．下線を引いた項は最小値を示し，その値はゼロであるとして計算してある)

余談になりますが，誰もが思う疑問「6116はどのタイミングにデータをメモリに書き込むのだろう?」ということについて考えてみます．6116のデータ・シートによりますと，「書き込みは$\overline{CS}$と$\overline{WE}$がともに"L"のときに行われる」と書いてあります．この文章は間違ってはいないのですが，$\overline{CS}$と$\overline{WE}$がともに"L"ということは，両信号が"L"になった時点，両信号が"L"になった中間時点，それに両信号が"L"である最後の時点のいずれをも考えることができます．

実際にはライト・データに対して，$\overline{CS}$または$\overline{WE}$が"H"になった時点からのセットアップ・タイム(t_{DW})とホールド・タイム(t_{DH})の規定がありますので，6116がデータをメモリに書き込む時点は，$\overline{CS}$または$\overline{WE}$が"H"になった時点と考えるのが妥当です．

I/Oとのインターフェース

ここではZ80とI/O(入出力ポートや周辺LSI)とのインターフェースについて説明をします．I/Oとメモリとの差異は平たくいってしまえば，データのやりとり制御に$\overline{IORQ}$信号が使われるか，$\overline{MREQ}$信号が使われるかにあるだけです．

I/Oの中でもっとも単純でかつ重要なのは，フリップフロップによる出力ポートと，3ステート・バッファによる入力ポートです．複雑な周辺LSIもCPUと面しているのは，これらの単純な入出力ポートであり，その先に複雑なロジック

がぶら下がっているといっても過言ではありません．

■フリップフロップによる出力ポート

図58にフリップフロップ，LS 273を用いた単純な出力ポートを示します．この回路のI/Oアドレス・デコーダは，A_3～A_7が"L"でかつ$\overline{IORQ}$信号が"L"のときにイネーブルされ，A_0～A_2をデコードし，ポート・セレクト信号($\overline{ポート00H}$～$\overline{ポート07H}$)を8個出力します．

LS273による出力ポートのアドレスは00Hであり，$\overline{ポート00H}$信号と$\overline{WR}$信号により，データ・バス上のデータをフリップフロップにラッチします．Z 80のI/Oライト・サイクルにおける$\overline{IORQ}$と$\overline{WR}$のタイミングはほとんど同じですので，この回路では$\overline{WR}$の立ち上がり時点に，データをラッチすることになります．

このポートにデータ55Hを書き込むときには，次のような命令を実行します．

```
LD     A, 55H
OUT   (00H), A
```

この命令により，データ55HがLS 273にラッチされ，LS 273のピン②，⑥，⑫，⑯が"H"になり，他の出力は"L"になります．

データをラッチするタイミングは，$\overline{WR}$の立ち下がり時点でもよいのですが，立ち上がり時点にしたほうがより安全であることを，図59により説明します．

I/Oライト・サイクルにおいて$\overline{WR}$信号の立ち下がり時点とデータ・バスの状態が確定する時点との時間間隔はt_{dci}というパラメータにより規定されており，Z 80 Aを4 MHzで動作させた場合の最悪値は－20 nsです．

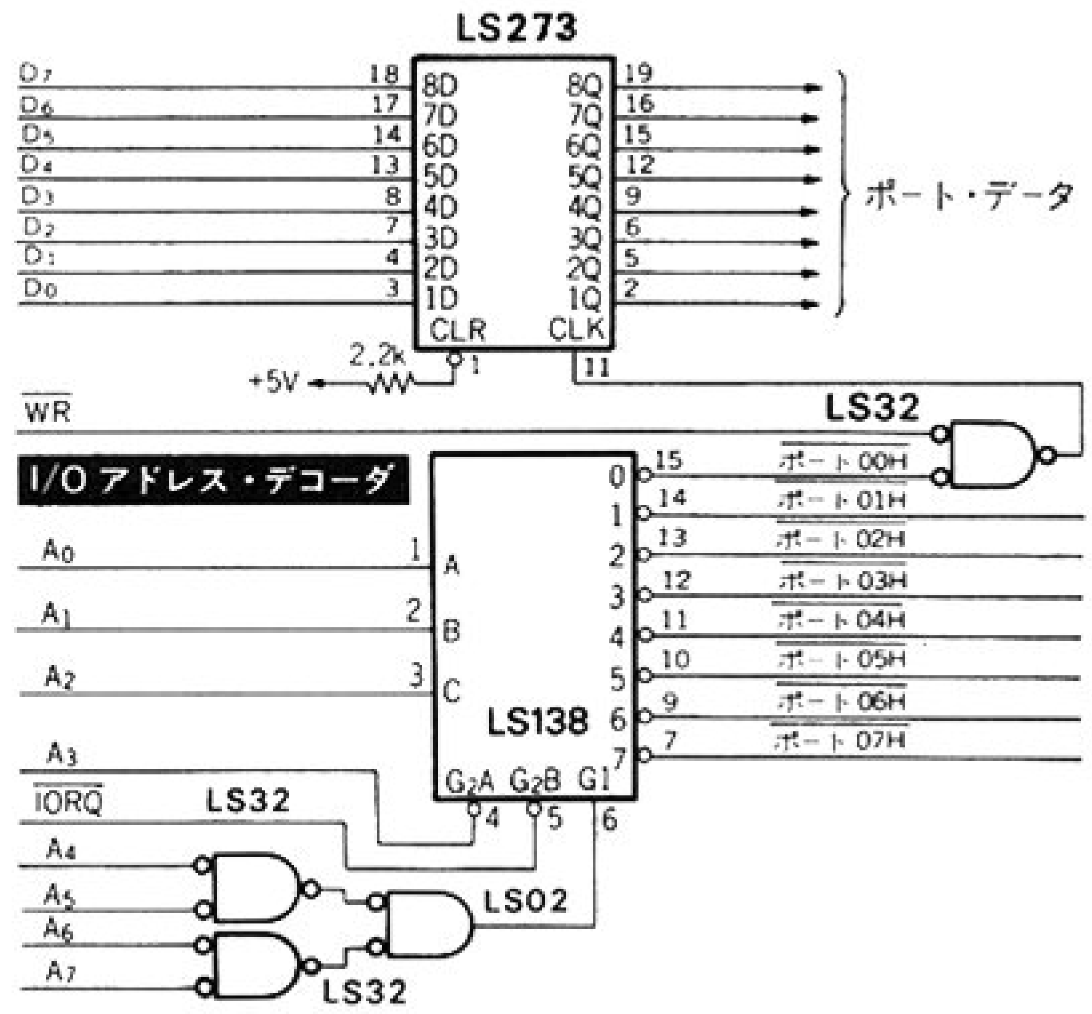
このことは最悪ケースの場合，データが確定するのは$\overline{WR}$の立ち下がりも20 ns遅れることがありうることを示しています．したがって，運の悪い場合にはデータが確定する前に$\overline{WR}$が"L"になることもありうるわけです．

またフリップフロップには，データ・セットアップ・タイムの規定があり，クロックよりデータが先に確定している必要があります．LS 273の場合，データ・セットアップ・タイムの最小値は20 nsです．

したがってこの時間をも考えに入れると，$\overline{WR}$の立ち下がりでフリップフロップをトリガするのは，ますます危険になってきます．

ところが$\overline{WR}$の立ち上がり時点にフリップフロップをトリガすれば絶対に安全です．図59からわかるように，$\overline{WR}$の立ち上がりのタイミングとデータが変化するタイミングは，たとえ$t_{F(D)}$をゼロと見積っても，決してオーバラップするこ

〈図58〉フリップフロップLS 273による出力ポート
(ポート・アドレスは00 H)



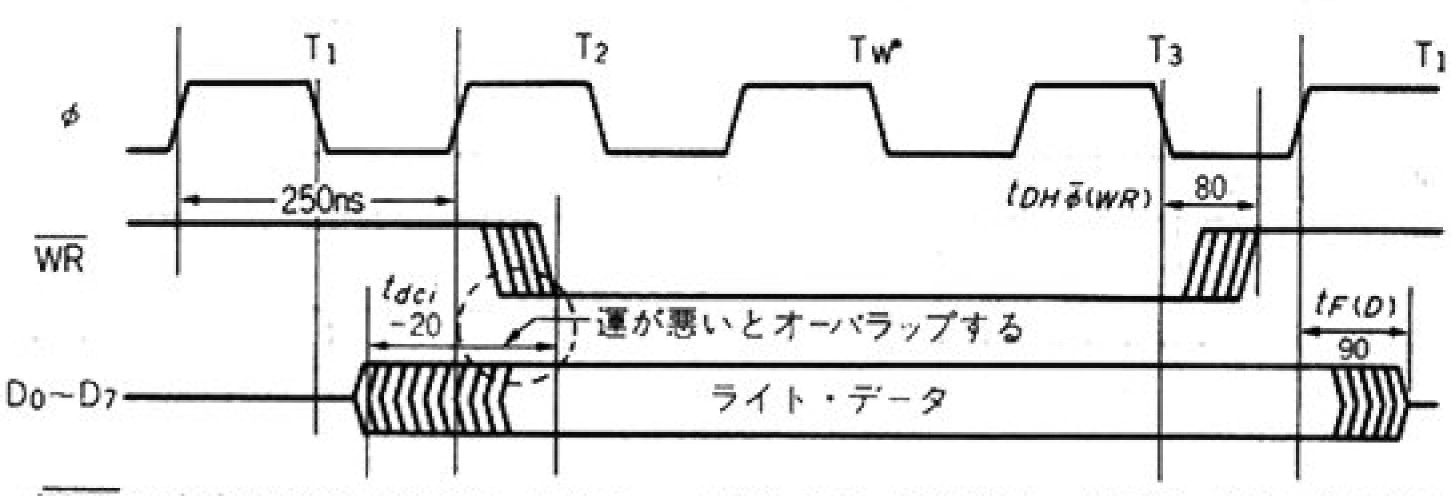
とはなく，その上フリップフロップのセットアップ・タイムも十分以上にあることがわかります．

この問題に限らず，ロジック設計をするときには常に最悪ケースを考慮しておかないと，使用したICのメーカーが異なったり，温度や電圧の変動により動かなくなったりすることがよくあります．

図58の回路では，I/Oアドレス・デコーダは，A_0～A_7をすべてデコードしているので，256個のI/Oアドレスをすべて識別できますが，場合によっては「オレはそんなにたくさんI/Oを使わない，8個もあれば十分だ」というようなこともよくあります．このようなときには，ややインチキではありますが，A_0～A_7の各アドレス・ビットに各I/Oポートを対応させることにより，I/Oアドレス・デコーダを単純にすることができます．図60にアドレス・ビットによるI/Oポートの選択を示します．

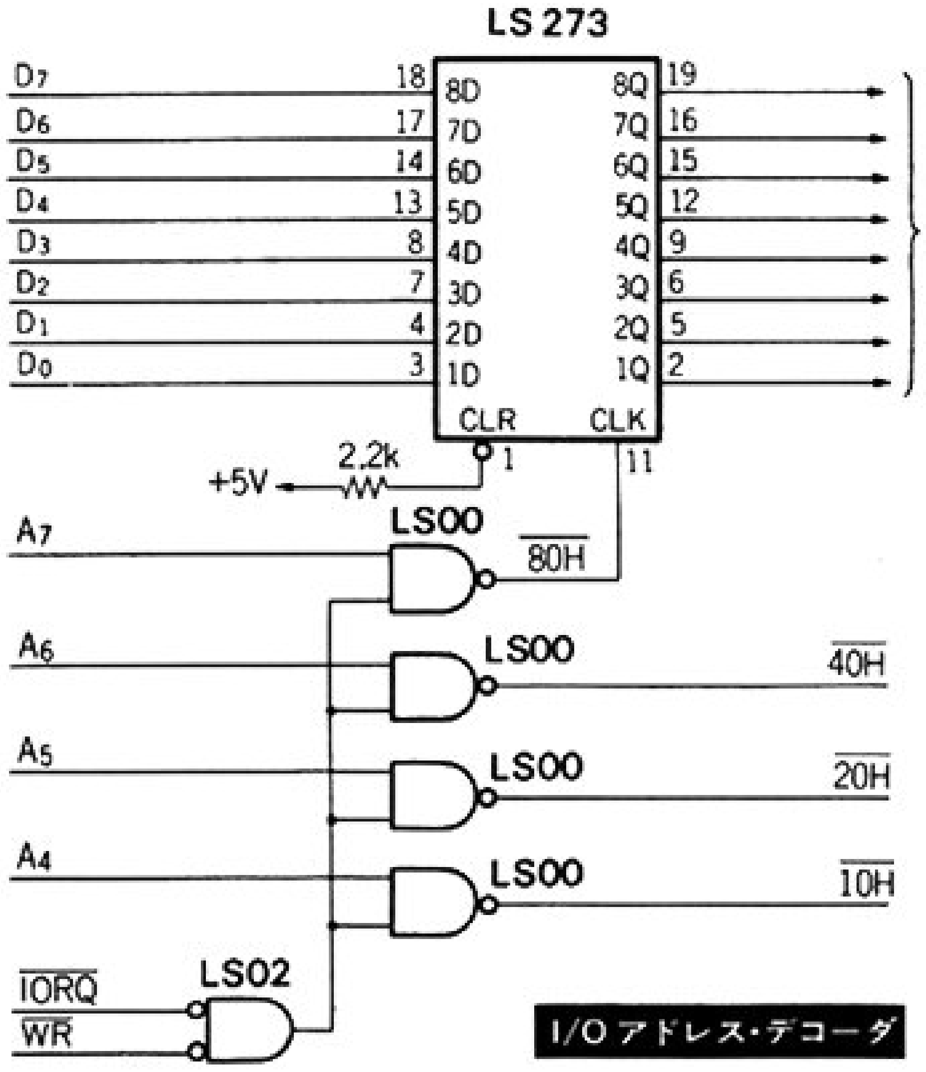
この方法ではI/Oアドレスとして，01 H，02 H，04 H・・・・40 H，80 Hのようにアドレス・ビットの中のどれか1ビットだけしか"1"にならないようなアドレス

〈図59〉I/Oライト・サイクルにおける$\overline{WR}$とデータのタイミング
(時間はZ 80 Aの場合で，単位はns)

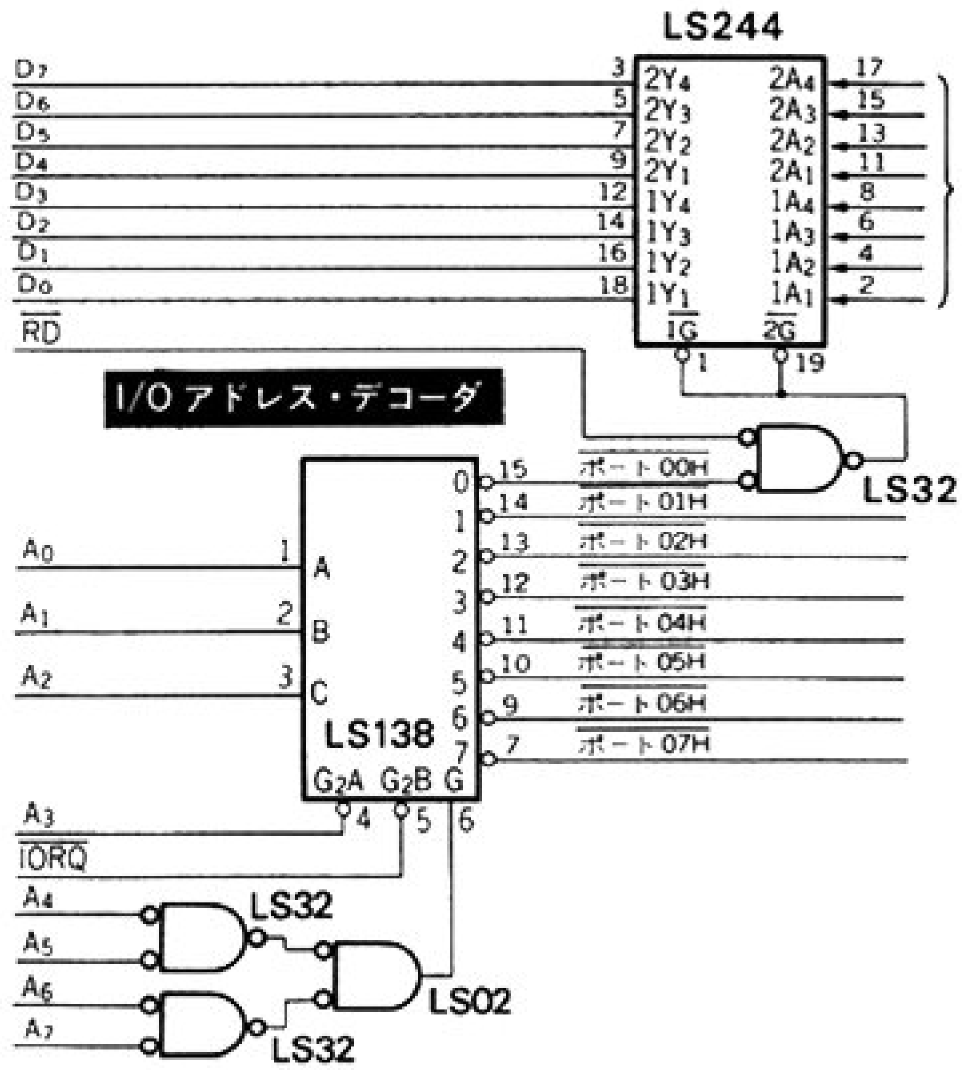


($\overline{WR}$の立ち下がりでフリップフロップをトリガすると，危険な場合がある)

〈図 60〉 アドレス・ビットによるポートの選択



〈図 61〉 LS 244 を用いた入力ポート



を用います．したがってアドレス・デコーダは他のアドレス・ビットの組み合わせに関知せず，A_7が“H”であればポート 80 Hを選択し，A_5が“H”であればポート 20 Hを選択します．

■ 3 ステート・バッファによる入力ポート

図 61 に 3 ステート・バッファLS 244 を用いた，入力ポートを示します．I/Oアドレス・デコーダの回路は図 57 の場合とまったく同じといってよいくらいで，異なるのは$\overline{WR}$の代わりに$\overline{RD}$が用いられている点です．この入力ポートのI/Oアドレスは 00 Hであり，このポートからデータを読み取るには次の命令を実行します．

IN　A,　(00 H)

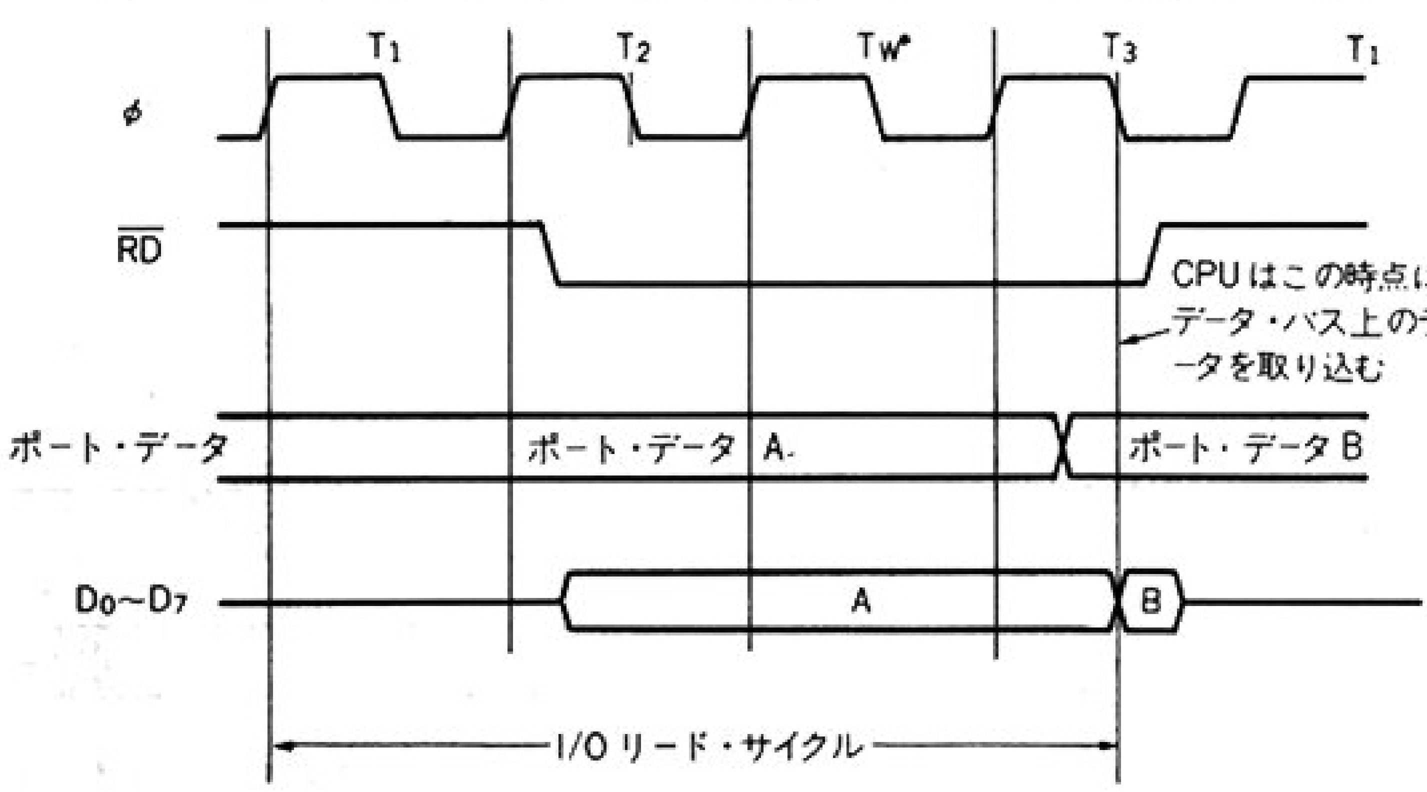
この命令により，LS 244 の入力ピン(②，④，⑥，⑧，…)の状態が，CPUのアキュムレータに読み込まれます．

図 61 の回路では，IN命令の実行中にLS 244 の入力ピンの状態が変化すると，正しいデータがCPUに届かない場合があります．図 62 によりこのケースを説明します．

I/Oリード・サイクルではZ 80 は，T_3のクロックの立ち下がり時点にデータ・バス上のデータをCPU内に取り込みますが，その時点の少し前にポート・データが変化すると，CPUがデータを取り込むタイミングにおけるデータ・バス上のデータは不確定となり，CPU内に取り込まれるデータは保証されません．

「I/Oリード・サイクル中には，ポート・データは絶対に変化しない」という保証のあるシステムでは，図 61 の入力ポートで十分なのですが，ポート・データがいつ変化するかわからないようなシステムでは，CPUがI/Oリード・サイクルを実行しているときには，たとえポート・データが変化しても，データ・バス上のデータを変化させないようなくふうが必要です．

〈図 62〉 I/Oリード・サイクル中に入力ポートのデータが変化した場合



これには 3 ステート・バッファの代わりに 3 ステート・バッファ付きのラッチを用います。図 63 にLS 373 を用いた入力ポートを示します．この回路では$\overline{RD}$信号によりポート・データをLS 373 内部のフリップフロップにラッチしてしまうので，$\overline{RD}$信号が“L”である期間は，ポート・データが変化しても，データ・バス上のデータは変化しません．

■ 16 ビットI/Oアドレスを用いる

いままでの説明では，I/Oアドレスは

A_0～A_7の8ビットであるとしてきましたが，Z80では16ビットのI/Oアドレスを使うことができます．入出力命令の中の，OUT (n)，AとIN A，(n)命令では，nの値(I/Oアドレス)がA_0～A_7に出力され，A_8～A_{15}にはアキュムレータの値が出力されます．

ところがZ80固有の入出力命令，OUT (C)，rやIN r，(C)命令では，レジスタCの値がA_0～A_7上に出力され，レジスタBの値がA_8～A_{15}上に出力されます．したがって，入出力命令としてこれらの命令だけを使う場合には，I/Oアドレスとして16ビットのアドレスを使うことができます．図64に16ビットのI/Oアドレスを用い，2Kバイトのメモリを I/Oとして使う例を示します．

図64ではスタティックRAM 6116はI/Oアドレス8000H～87FFH番地の範囲にあり，8000H番地にデータ55Hを書き込む場合には，

```
LD   BC, 8000H
LD   A, 55H
OUT  (C), A
```

とすればよく，87FF番地のデータを読むには，

```
LD   BC, 87FFH
IN   A, (C)
```

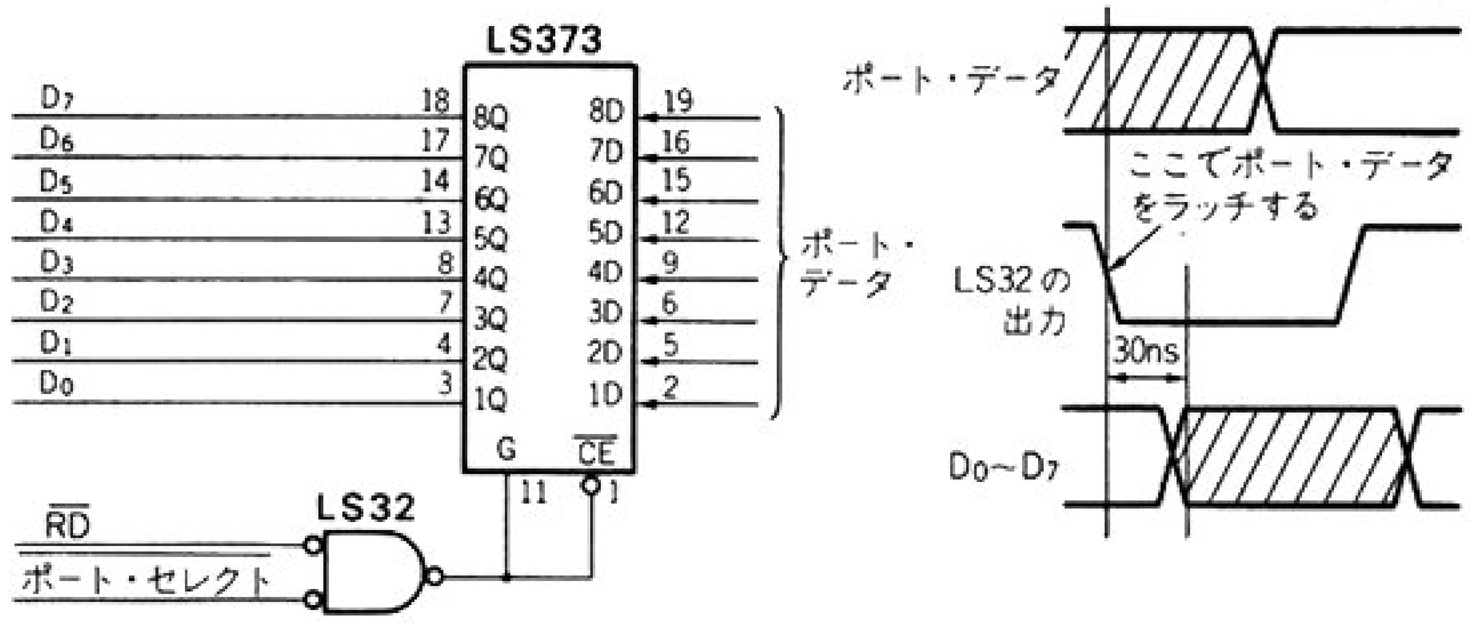
とすればよいわけです．

このように16ビットのI/Oアドレスを使用した場合には，レジスタBの値が変化するような入出力命令，たとえばOTIRやINIRなどは使えません．これらの命令ではレジスタBはデータ・レングス・カウンタとして使われますので，1バイトのデータを転送するたびに，1だけ減じられるからです．

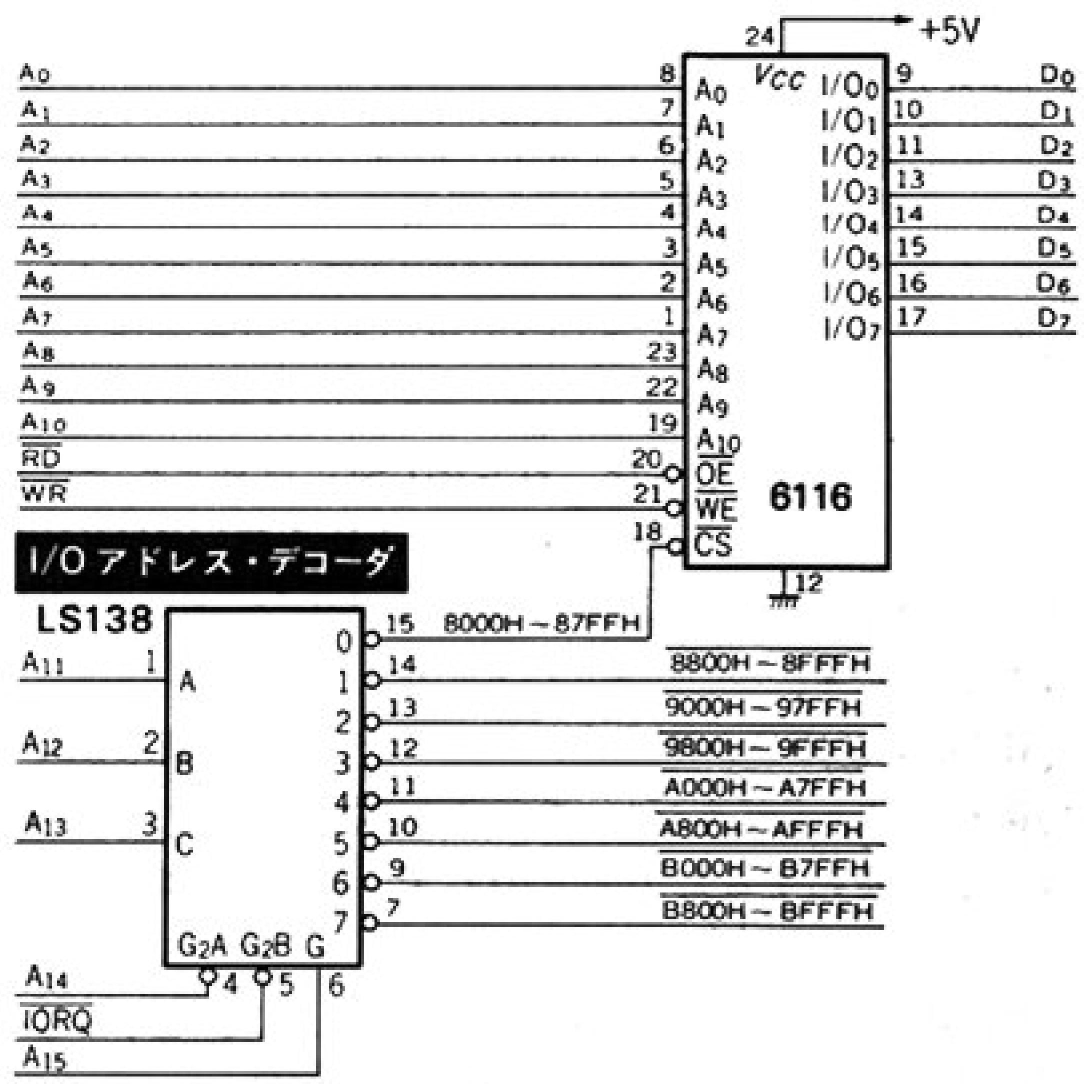
16ビット・アドレスを使用するという考えは，Z80 DMAを使うときにも有効です．Z80 DMAは16ビットのI/Oアドレスを出力できます．したがってDMAデータ転送により，大量のメモリ・データをI/O空間のバッファ・メモリに転送することができます．このような応用としては，CRTリフレッシュ回路のリフレッシュ・バッファなどがあげられます．

この場合には16ビットのI/O空間は，DMAデータ転送でしかデータのやりとりをしない，というような条件を付ければ(I/Oアドレス・デコーダの条件に$\overline{BUSAK}$を追加する)，入出力命令におけるA_8～A_{15}の値はどんな値になってもよく，8ビットI/Oアドレスの I/Oと混在が可能です．

〈図63〉ラッチ付き入力ポート．I/Oリード・サイクル中にポート・データが変化しても，データ・バスの状態は変らない



〈図64〉16ビットI/Oアドレスを用いたI/O空間のRAM



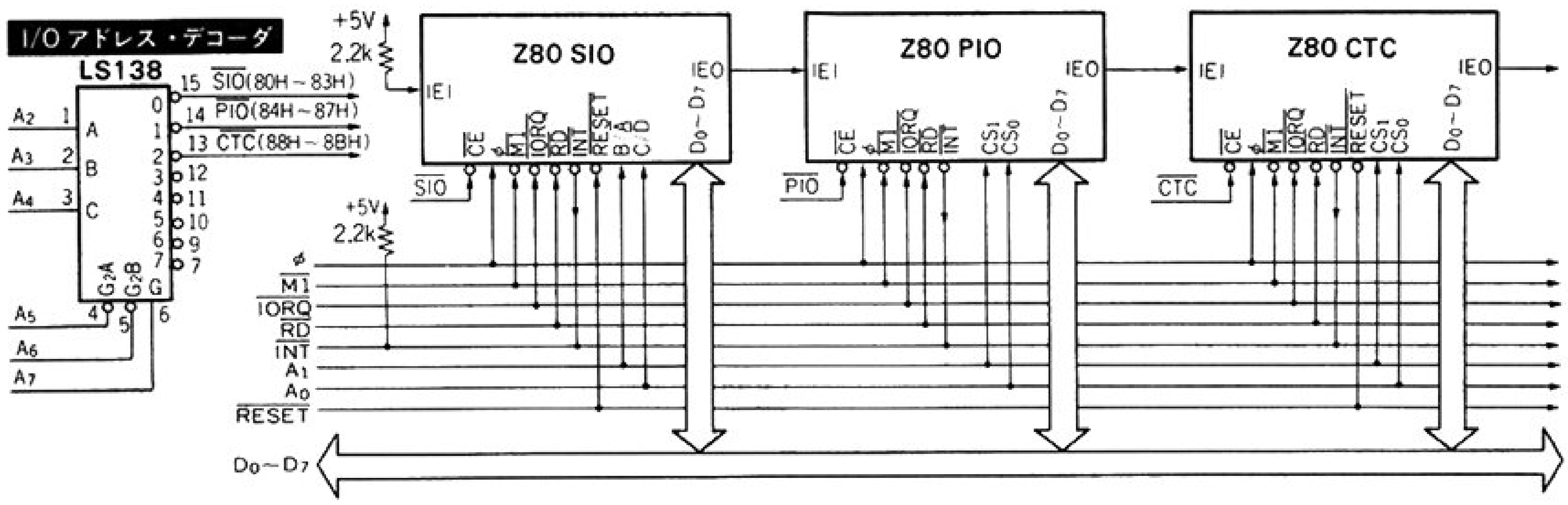
■Z80ファミリの周辺LSIとのインターフェース

単純な入出力ポートとのインターフェースもだいじですが，Z80 CPUを使う際には，Z80ファミリの周辺LSI(PIO, CTC, SIO, DMAなど)とのインターフェースもたいせつです．Z80ファミリの周辺LSIはかなり高度の機能を持っていますので，これらを使わないという手はありません．

図65にZ80ファミリの周辺LSIである，SIO，PIO，CTCとのインターフェースを示します．DMAも周辺LSIの中の一つですが，これはI/Oというよりもデータ転送用のバス・コントローラですので，ここでは取り扱いません．

図65からわかるように，これら三つの周辺LSIとZ80とのインターフェースはほとんど同じである，ということができます．異なる点があるとすれば，PIOには$\overline{RESET}$ピンがないくらいなものです．これらのLSIはすべて$\overline{IORQ}$ピンがあるので，I/Oアドレス・デコーダはアドレスだけをデコードします．またアドレスは通常上位6ビットだけをデコードし，下位2ビット

〈図 65〉 周辺LSIとのインターフェース



(A_0 と A_1) は直接LSIに接続し，これらの2ビットによりLSI内部のポートを選択します．

これらのLSIにはすべて$\overline{\text{M1}}$信号が接続されていますが，これはZ80の割り込み認識サイクルを可能にするためです．割り込み要求線($\overline{\text{INT}}$)は，ワイヤードOR接続ができるように，各LSIの出力型式がオープン・ドレイン型になっていますので，+5Vにプル・アップする必要があります．またこれらのLSIには$\overline{\text{WR}}$ピンはなく，$\overline{\text{RD}}$ピンだけしかありません．これらのLSIは$\overline{\text{IORQ}}$が"L"で$\overline{\text{RD}}$が"H"のときには$\overline{\text{WR}}$が"L"であると解釈するからです．

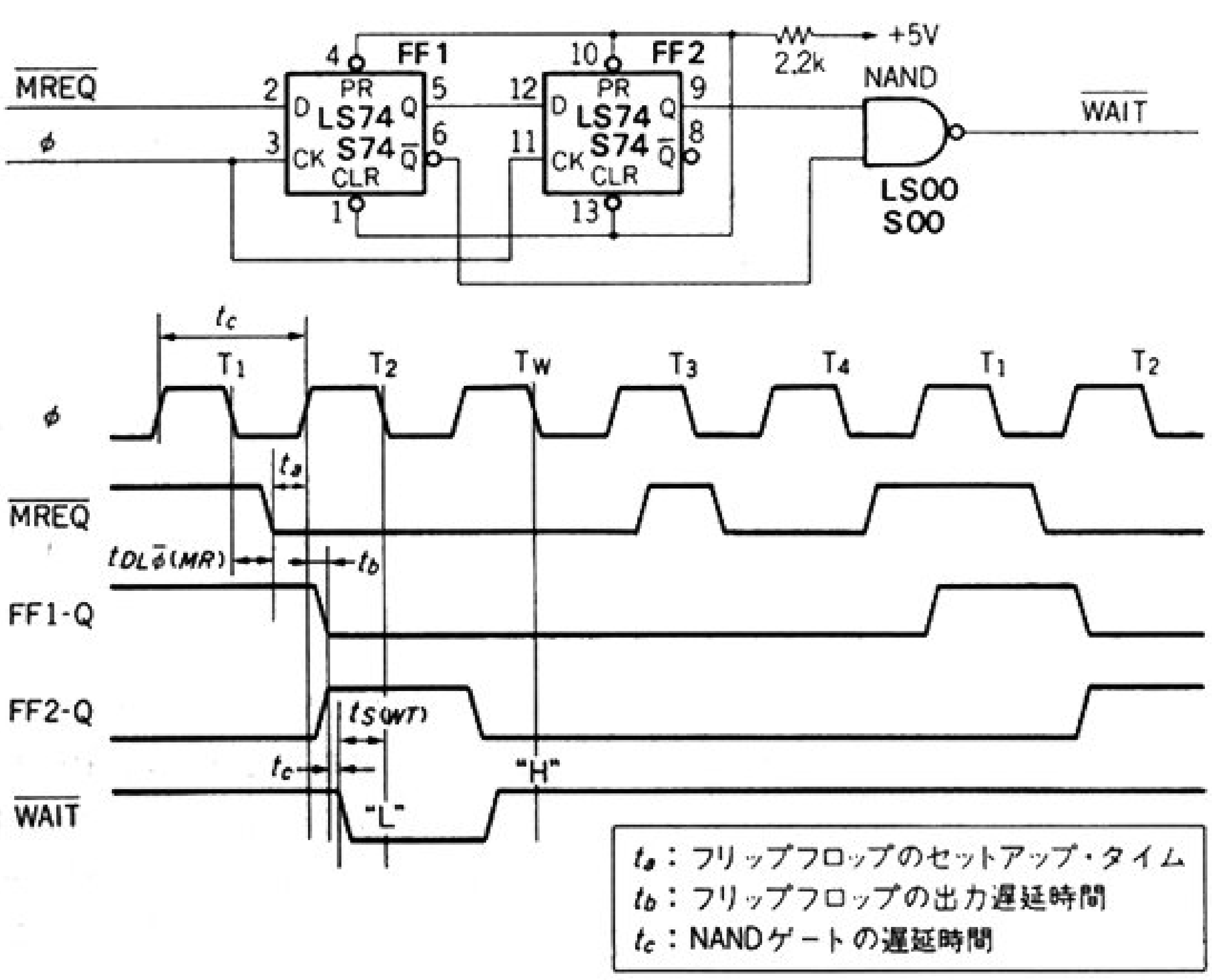
IEIおよびIEOの接続は，Z80ファミリの周辺LSIの特徴をなす，「デイジー・チェイン式の優先割り込み制御」を可能にするためのものです．図65の接続では，SIOの割り込み優先度が一番高く，CTCが一番低くなっています．これらのことがらに関する詳しい説明は，「Z80周辺LSIの割り込み制御について」をごらんください(361ページ)．

■$\overline{\text{WAIT}}$信号の使いかた

$\overline{\text{WAIT}}$信号の機能は元もとCPUの速度に比較して遅いメモリやI/Oサイクルを引き延ばしするためのものでした．しかしながら現在入手できるメモリやI/Oは十分に高速なので，こうした目的のために$\overline{\text{WAIT}}$信号を用いることは，ごくまれにしかありません．

ごくまれに$\overline{\text{WAIT}}$信号が用いられるケースとしては，CRTリフレッシュ回路がCPUにウェイトをかけて表示用のデータをメモリから読み取る場合や，割り込

〈図 66〉 メモリ・サイクル(M1サイクル，メモリ・リード/ライト・サイクル)に1ウェイト・ステートを挿入する回路



〈図 67〉 図66の回路の時間的定数の値とTTLの遅延時間(単位：ns)

記号＼タイプ	Z80	Z80A	Z80B
t_c	400	250	165
$t_{DL\overline{\phi}(MR)}$	100	85	70
t_a	100	40	12.5
$t_{s(WT)}$	70	70	60
必要な t_b+t_c の値	130以下	55以下	22.5以下

	パラメータ＼ICタイプ	LSタイプ	Sタイプ
t_a	FFのセットアップ・タイプ	20	3
t_b	FFの出力遅延時間	40	9
t_c	NANDゲートの遅延時間	15	5
	t_b+t_c	55	14

FF：LS74/S74，NANDゲート LS00/S00

この図からZ80Bを6MHzで動作させるときには，Sタイプの TTLを用いる必要があることがわかる．

みデイジー・チェインの伝搬時間を稼ぐために，ウェイト・ステートを用いて，割り込み認識サイクルを引き延ばす場合などがあげられます．

$\overline{\text{WAIT}}$信号が用いられることが少なくなったとはいえ，$\overline{\text{WAIT}}$信号の使いかたそのものは基本テクニックであり，使うと使わないとにかかわらず，だれもが知っていなければならない技法です．

■メモリ・サイクルにおけるウェイト

図66にメモリ・サイクルに1ウェイト・ステートを挿入するための回路を示します．この図にはM1サイクルにおけるタイミングが示されていますが，この回路はメモリ・リード/ライト・サイクルにおいても動作します．

この回路は$\overline{\text{MREQ}}$が“L”になると，その直後のクロックの立ち上がり時点にFF_1のQが“L”になり，その次のクロックの立ち上がり時点にFF_2のQが“L”になります．$\overline{\text{WAIT}}$信号はFF_1のQが“L”でFF 2のQが“H”である1クロック期間だけ“L”になります．

またこの図には各信号の遅延時間，およびセットアップ・タイムが記入されています．この回路がきちんと動作するための時間的要素は，FF_1のデータ・セットアップ・タイム(t_a)，FF_1の出力遅延時間(t_b)，NANDゲートの遅延時間(t_c)ならびにT_2のクロックの立ち下がり時点に対する$\overline{\text{WAIT}}$信号のセットアップ・タイム($t_{S(WT)}$)です．

図67にZ80，Z80A，Z80Bをそれぞれ最高速度で動作させた場合の各時間の最悪値を示します．またこの図にはLSタイプ，およびSタイプのフリップフロップ(LS74とS74)とNANDゲート(LS00とS00)の時間的定数も示してあります．

この図から，フリップフロップのデータ・セットアップ・タイムについては，Z80およびZ80AではLS74で十分であり，Z80Bの場合にはS74を使わないといけないことがわかります．また$\overline{\text{WAIT}}$信号のセットアップ・タイムを満足させるには，フリップフロップとNANDゲートの遅延時間の和，t_b+t_cの値は，Z80，Z80A，Z80Bの場合，それぞれ130ns，55ns，22.5nsとなります．

このt_b+t_cの値は，LSタイプのTTLを用いたときには55nsであり，Sタイプのときには14nsになります．したがって，Z80およびZ80Aの場合にはLSタイプのTTLで間に合い，Z80Bの場合にはSタイプのTTLを用いなければなりません．

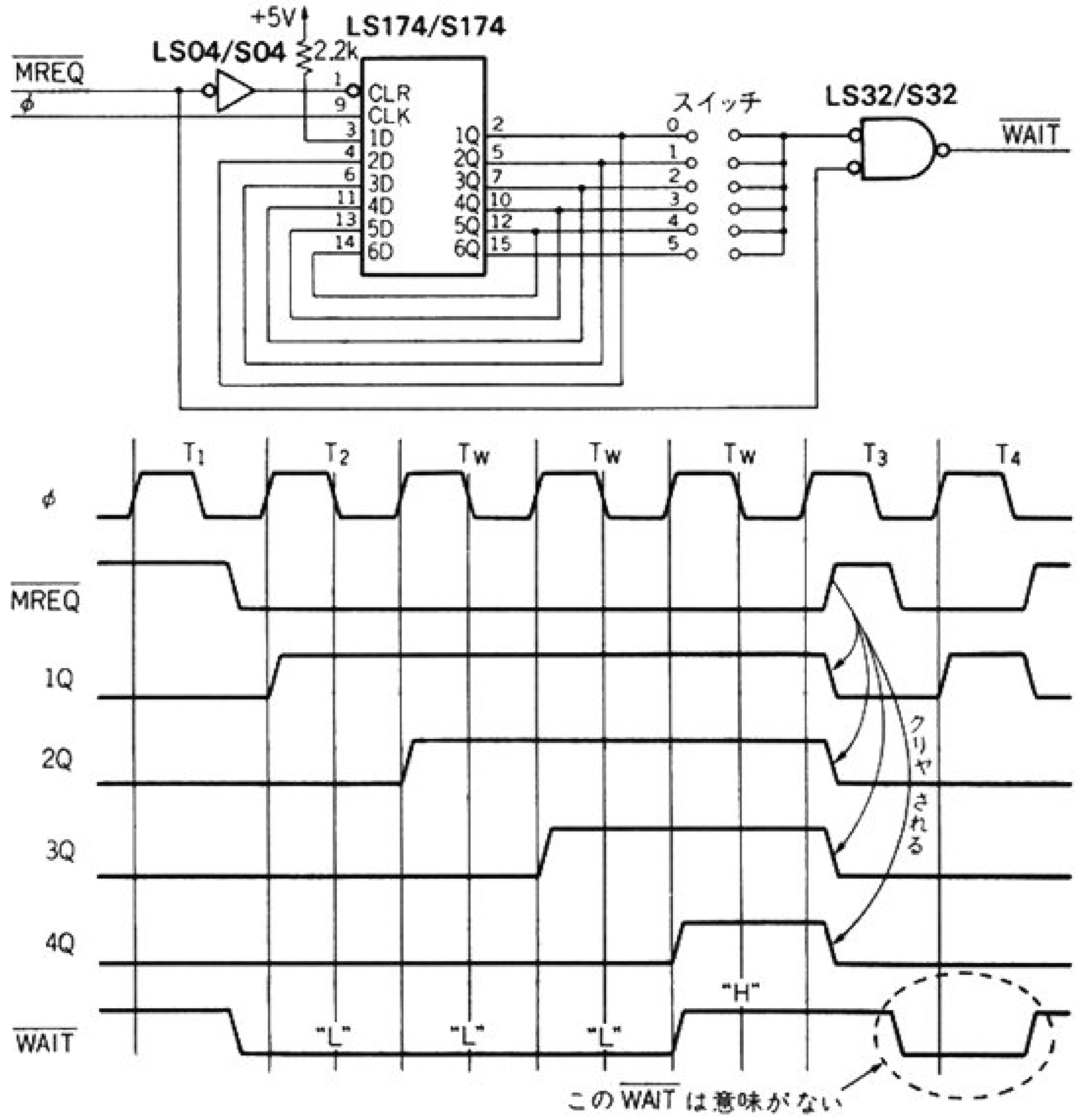
このようにCPU周辺の回路を設計する際には，常に時間に気を配っておく必要があります．図66のような非常に簡単な回路でさえ，LSタイプのTTLを使うか，SタイプのTTLを使うかを，きちんと計算して決める必要があります．

図66の回路では$\overline{\text{MREQ}}$により無条件に$\overline{\text{WAIT}}$信号を出力していますが，特定のメモリ・アドレスをアクセスする場合だけ，ウェイト・ステートを挿入したいような場合には，$\overline{\text{MREQ}}$とメモリ・アドレス・デコーダの出力の論理積を$\overline{\text{MREQ}}$の代わりに使います．またM1サイクルだけでウェイト・ステートを挿入したい場合には，$\overline{\text{MREQ}}$と$\overline{\text{M1}}$の論理積を使えばよいことになります．

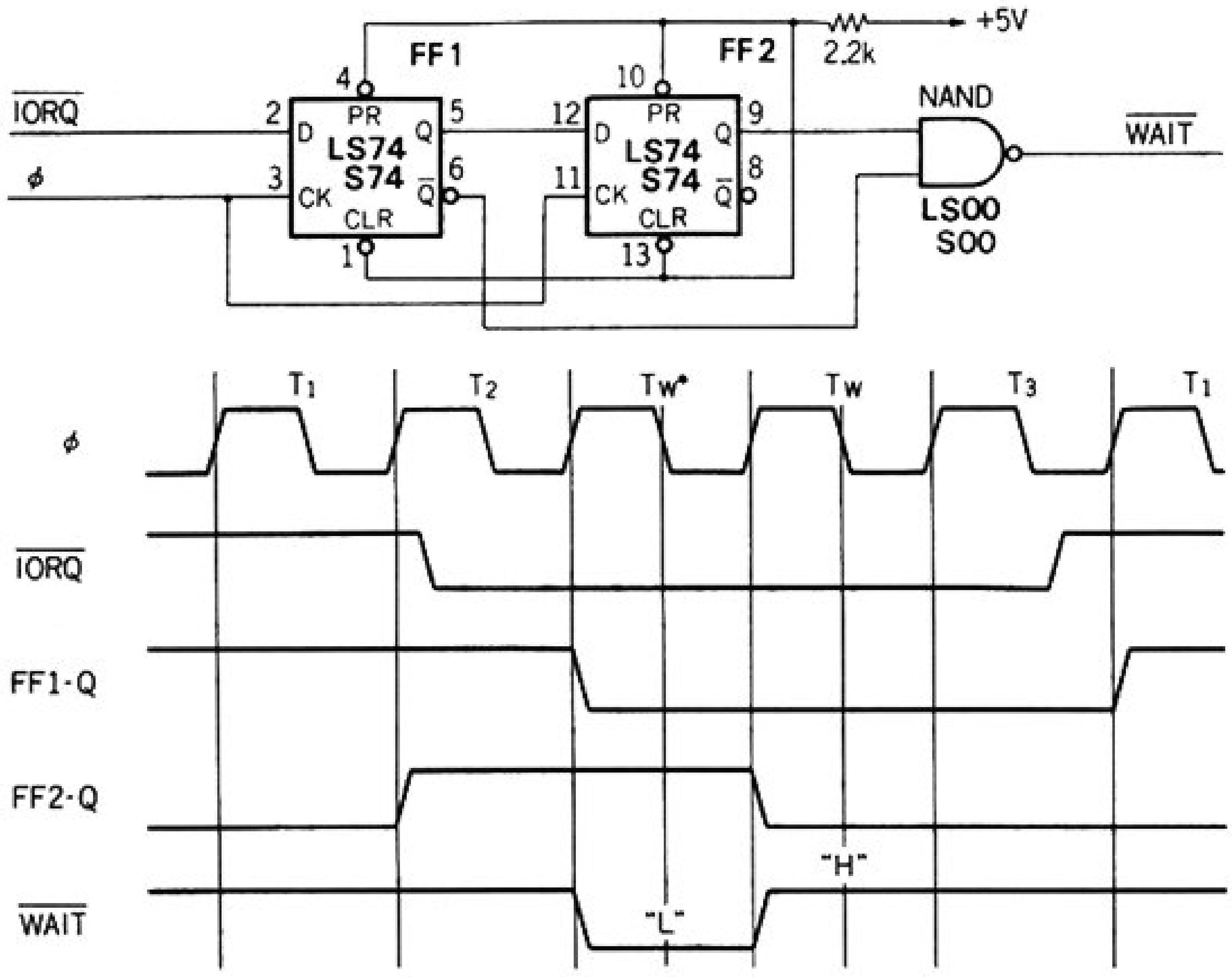
図68にスイッチの切り替えにより，0〜5個のウェイト・ステートを挿入する回路を示します．この回路では，$\overline{\text{MREQ}}$が“H”のときには174はクリヤされています．$\overline{\text{MREQ}}$が“L”になると，174はクリヤ状態は解除され，その後のクロックの立ち上がり時点に，174の出力1Qが“H”になり，以降のクロックの立ち上がり時点に，2Q〜6Qが順に“H”になっていきます．

この回路では，スイッチのどこをショートするかに

〈図68〉メモリ・サイクルに0〜5個のウェイト・ステートを挿入する回路．図のタイミング・チャートは，スイッチ3をショートした場合の例



〈図69〉I/Oサイクルに1ウェイト・ステートを挿入する回路とそのタイミング



より，$\overline{\text{MREQ}}$が"L"になってから$\overline{\text{WAIT}}$が"H"になるまでのクロック数を，0～5の範囲で設定できます．図68には，3クロック期間の$\overline{\text{WAIT}}$信号を出力する場合のタイミングが示されています．

■I/Oサイクルにウェイト・ステートを挿入する

図69にI/Oサイクル中にウェイト・ステートを一つ追加するための回路と，その動作タイミングを示します．この回路は，図66の$\overline{\text{MREQ}}$を$\overline{\text{IORQ}}$に変えただけで，その他はまったく同一です．しかしながら，$\overline{\text{MREQ}}$が"L"になるタイミングと，$\overline{\text{IORQ}}$が"L"になるタイミングは約半クロック・サイクルだけズレていますので，回路の動作タイミングは異なります．

図66の回路はT_2のクロックの立ち下がり時点に，$\overline{\text{WAIT}}$信号が"L"になるようになっていましたが，図69では$T_W{}^*$のクロックの立ち下がり時点に"L"になるようになっています．

まったく同じ回路が異なるタイミングで動作が可能ということは，図66または図69の回路がすばらしいというよりは，Z80のタイミングが綿密に設計されているからというべきでしょう．したがって，これらの回路のFF_1の入力として，$\overline{\text{MREQ}}$と$\overline{\text{IORQ}}$の論理和を用いれば，メモリ・サイクルにおいてもI/Oサイクルにおいても動作可能です．

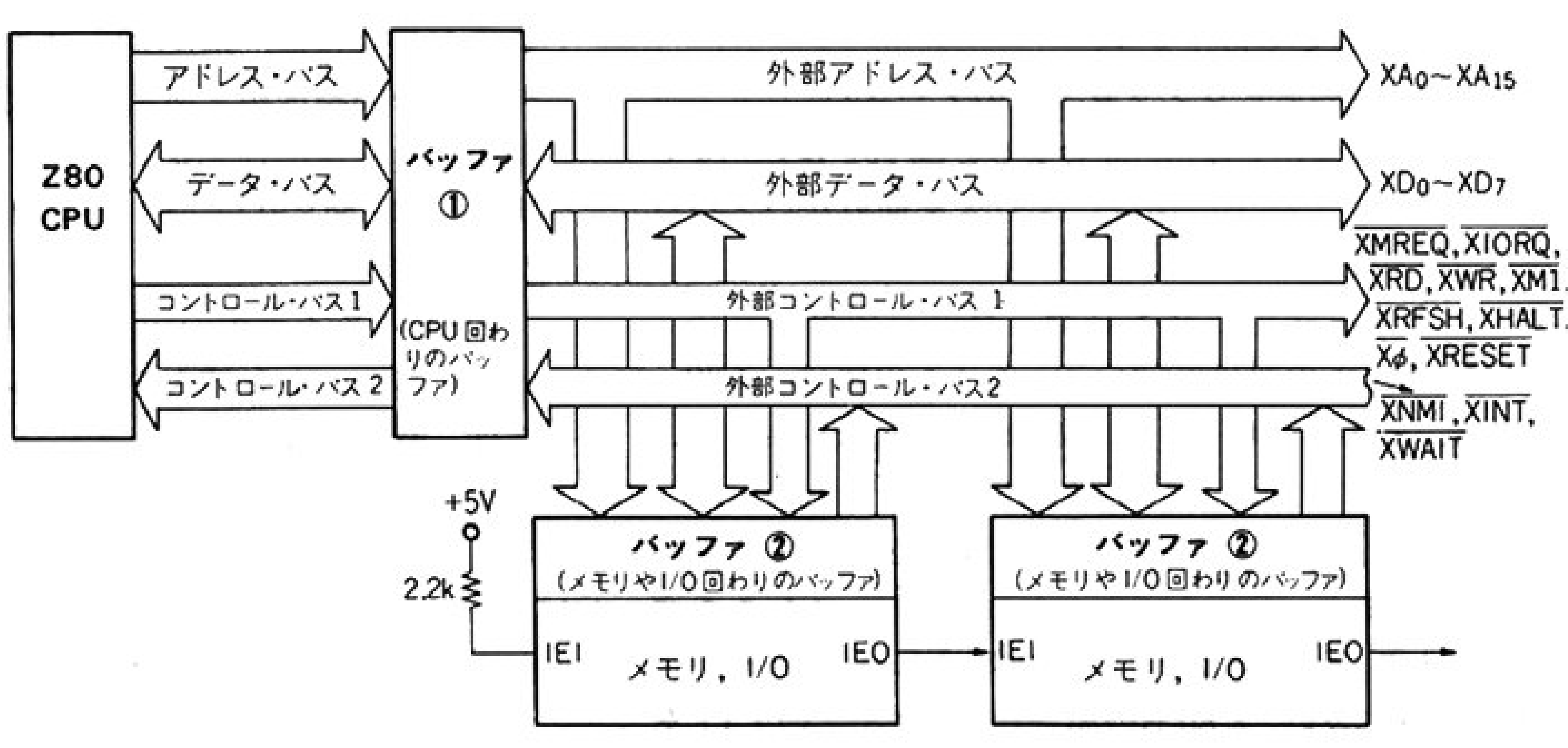
■バスのバッファリングについて

Z80の出力信号のファン・アウトは約2mAであり，SタイプのTTLなら1個，LSタイプのTTLなら5個の入力をドライブできます．したがってZ80の出力信号に，あまり多くのTTLがぶら下がるようなことのない小さなシステムでは問題になりませんが，比較的規模の大きなシステムでは，バスのバッファリングを行うのがふつうです．

図70にバス・バッファリングの概念図を示します．この図からわかるように，バス・バッファリングには二つ区別があります．その一つはCPU回わりのバッファであり(図70のバッファ①)，他はメモリやI/O回わりのバッファ(図70のバッファ②)です．

図71にCPU回わりのバッファを示します．アドレス・バスとコントロール・バス1は，LS244によりバッファリングされており，データ・バスは双方向性のドライバ，LS245により，バッファリングされています．また入力信号である外部コントロール・バス2の

〈図70〉
バスのバッファリング



信号は，+5Vにプルアップされており，オープン・コレクタのドライバにより，ワイヤードOR接続ができるようになっています．LS 244 やLS 245 を使用していたのでは，信号の遅延時間が問題になるような場合には，74 FシリーズのTTLを使えばよいと思います．

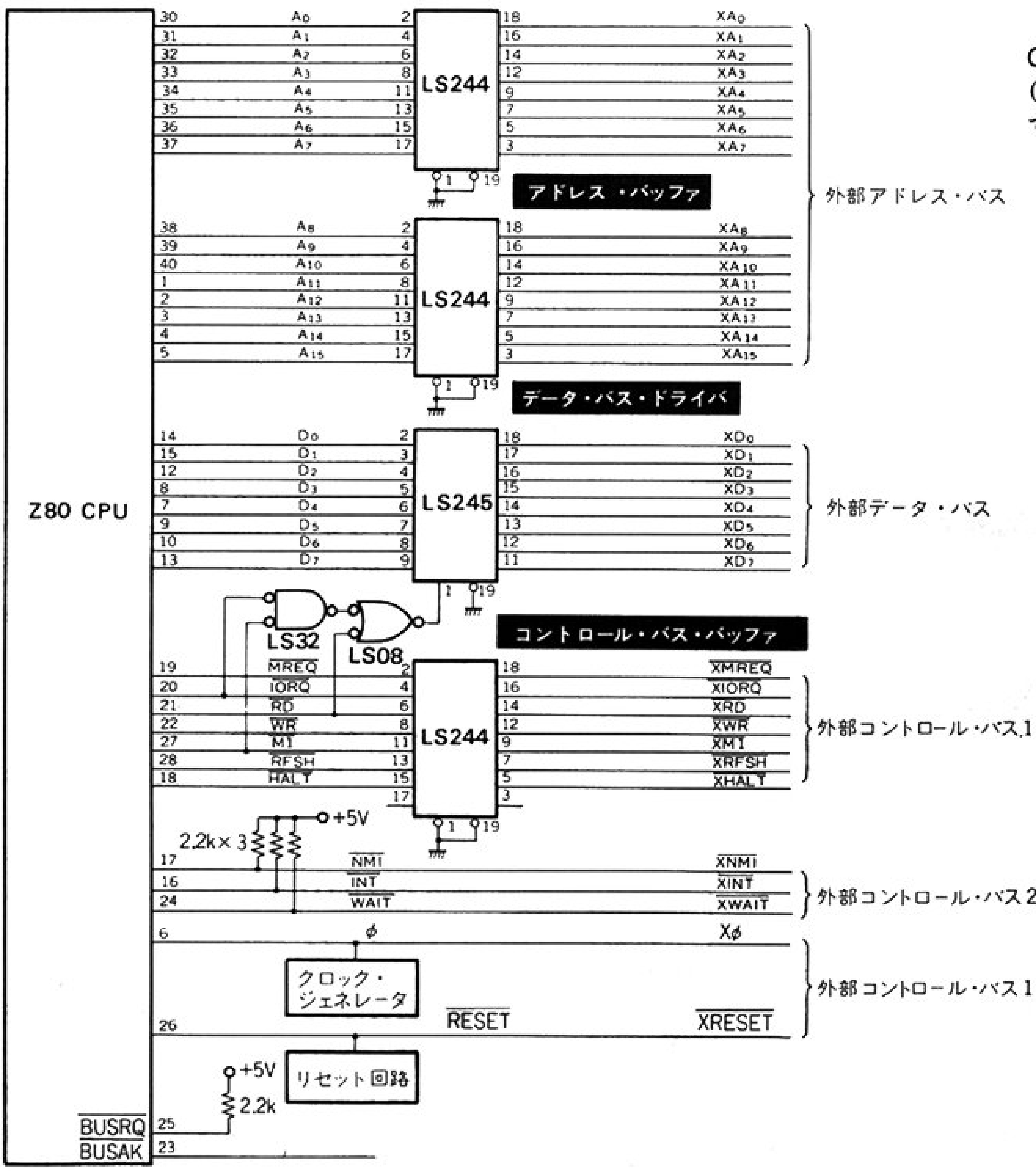
図71の回路の中で注目すべきことは，LS 245 の方向制御ピン(ピン①)に接続された二つのゲートです．LS 245 のピン①が“H”だと，データはCPU→外部の方向に伝達され，“L”のときには外部→CPUの向きに伝達されます．したがってふつうに考えれば，LS 245 のピン①には$\overline{RD}$信号を接続すればよいはずですが，そうしたのでは外部から割り込みベクトルを読むことができなくなります．

すでに述べたように，Z 80 の割り込み認識サイクルでは，$\overline{M1}$信号と$\overline{IORQ}$信号をともに“L”にして，CPUは割り込みベクトルを読み取ります．したがって，これらの信号がともに“L”のときにも，データ・バス・ドライバ(LS 245)の方向を，外部→CPUの向きにする必要があります．二つのゲートは$\overline{RD}$信号が“L”の場合，および$\overline{IORQ}$と$\overline{M1}$が“L”のときに，データ・バス・ドライバの方向を外部→CPUの向きにする働きをしています．

これとは逆に，LS 245 の入出力ピンを入れ替えて(ピン②↔ピン⑱など)，$\overline{WR}$信号によりデータ・バス・ドライバの向きを制御する方法も考えられますが，このようにすると，$\overline{WR}$信号が“L”になるまで，CPUの出力データは外部データ・バス上に乗らないことになるのでよい方法であるとはいえません．

図71の回路では，データ・バス・ドライバは通常，CPU→外部の方向に向いており，$\overline{RD}$信号が“L”になったとき，および割り込み認識サイクルにおいてのみ，外部→CPUの向きに変わります．

図71の回路ではDMAデータ転送を完全に無視してきましたが，この回路に対して何の変更も加えずに



〈図71〉
CPU回わりのバッファ回路
(図70のバッファ①に相当する)

DMAデータ転送が可能です．図72にDMAデータ転送を考慮した場合の構成を示します．図71の回路にZ 80 DMAを追加し，DMAとCPUの間に$\overline{\text{BUSRQ}}$信号と$\overline{\text{BUSAK}}$信号を接続します．そしてDMAリクエストのために，RDY信号を外部コントロール・バス2に追加します．

Z 80 DMAはCPUに対して$\overline{\text{BUSRQ}}$を出し，CPUから$\overline{\text{BUSAK}}$をもらうと，アドレス・バス，データ・バス，ならびにコントロール・バスをCPUとまったく同じように制御して，DMAデータ転送を行います．したがって外部バスの先に接がっているメモリやI/OはCPUによりデータ転送をしているのか，DMAによりデータ転送をしているかについてまったく気にする必要がありません．

〈図72〉 DMAを考慮した場合のバッファリング

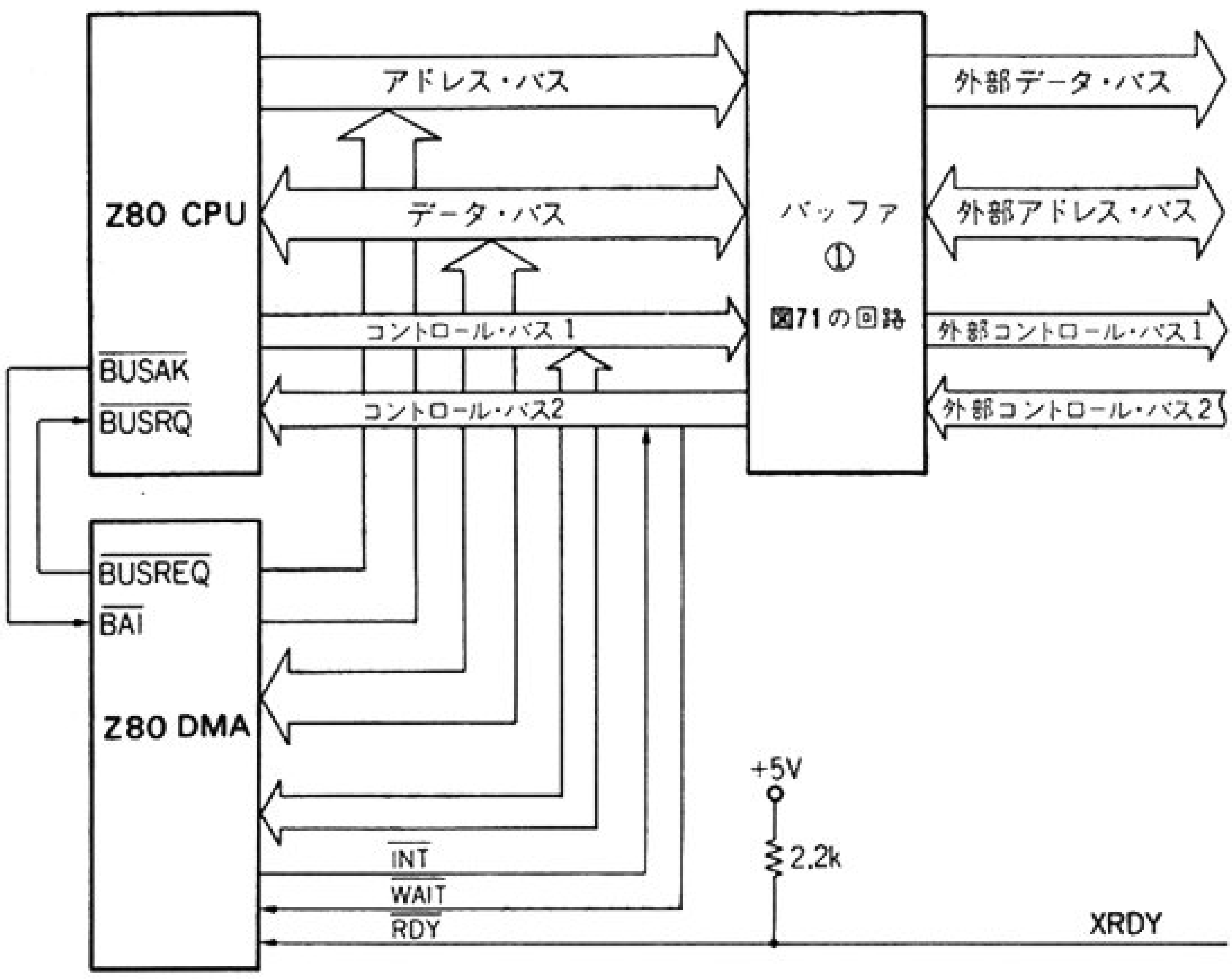


図73にメモリやI/Oの回わりのバス・バッファの回路を示します．この回路の中のアドレス・バッファとコントロール・バス・バッファは，図71の回路の場合とまったく同じです．$\overline{\text{NMI}}$，$\overline{\text{INT}}$それに$\overline{\text{WAIT}}$などは通常ワイヤードOR接続されるので，オープン・コレクタのドライバ7406により駆動しています．

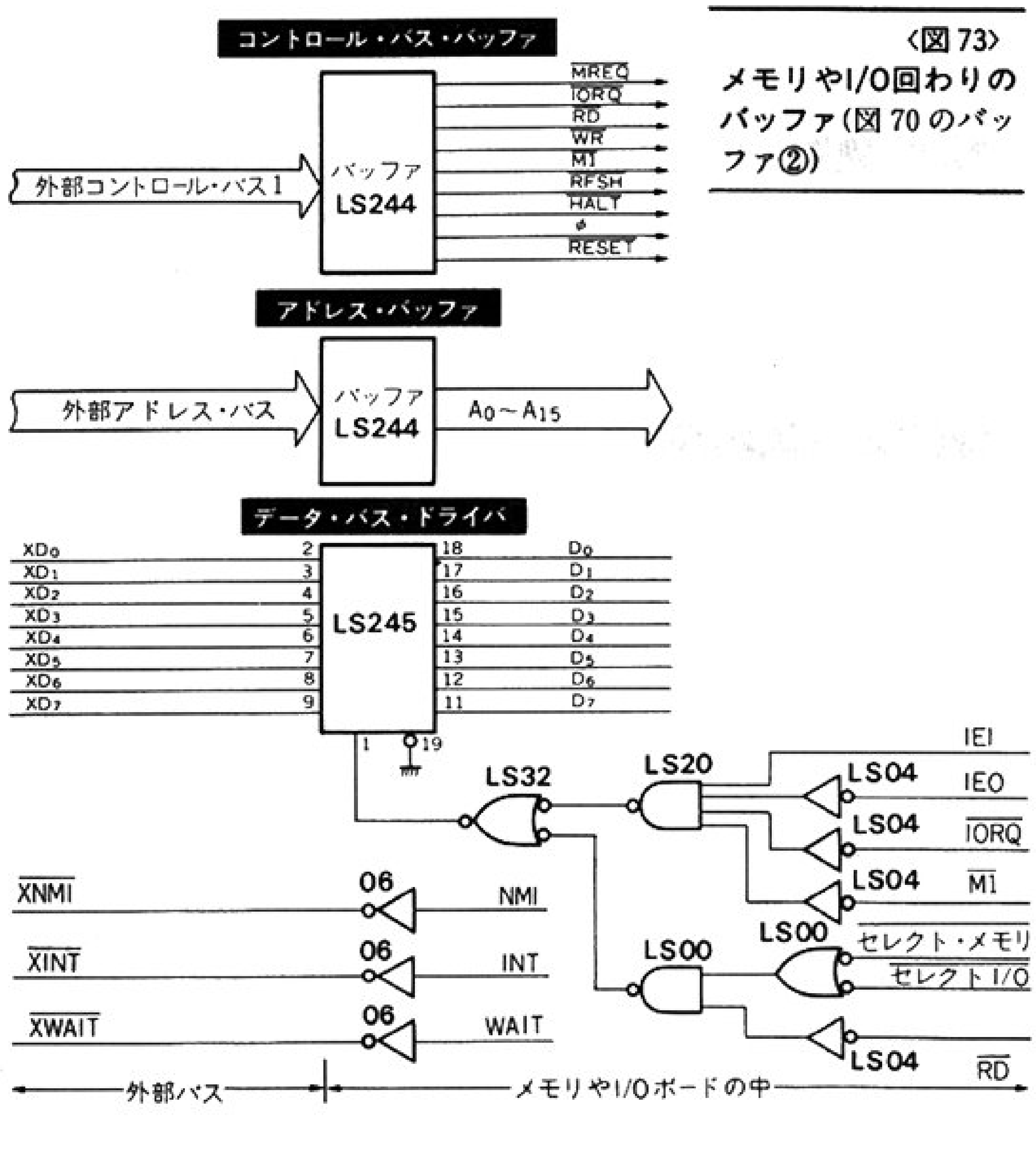
この回路の中で注意すべき点は，データ・バス・ドライバの方向制御(LS 245のピン①)です．LS 245のピン①が“L”であると，データはピン⑱→ピン②の向きにドライブされます．この向きにLS 245がドライブされるのは，

① メモリやI/Oがセレクトされ，かつ$\overline{\text{RD}}$信号が“L”のとき

② 割り込みベクトルが読み込まれるとき

です．①は$\overline{\text{セレクト・メモリ}}$，または$\overline{\text{セレクトI/O}}$が“L”で，かつ$\overline{\text{RD}}$が“L”のときに，LS 32の一方の入力が“L”になることを意味します．②に関しては，IEIが“H”で$\overline{\text{IEO}}$が“L”のときに，$\overline{\text{IORQ}}$および$\overline{\text{M1}}$がともに“L”(割り込み認識サイクル)の場合に，LS 32のもう一つの入力が“L”になることにより制御されています．

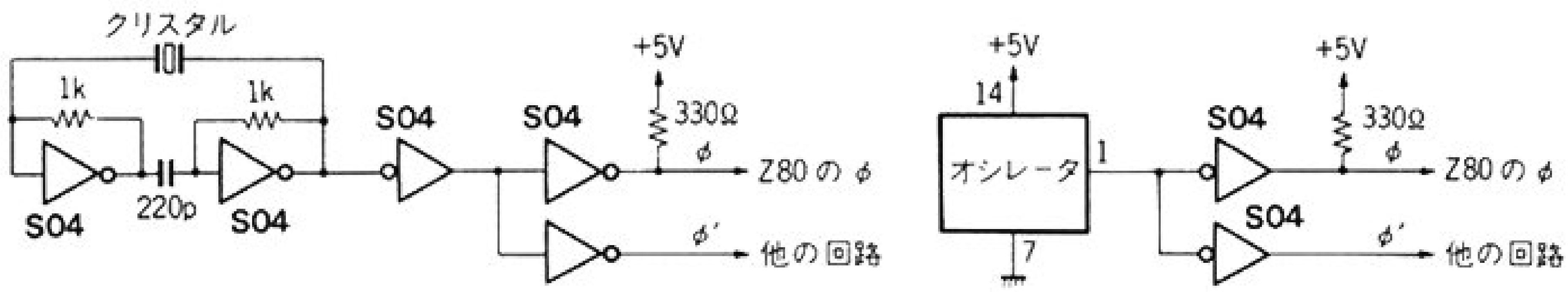
Z 80における優先割り込み制御用のデイジー・チェインは，IEIが“H”である割り込みソースだけが割り込みを発生することができ，割り込みを要求している場合にはIEOを“L”にすることになっているからです．



〈図73〉 メモリやI/O回わりのバッファ(図70のバッファ②)

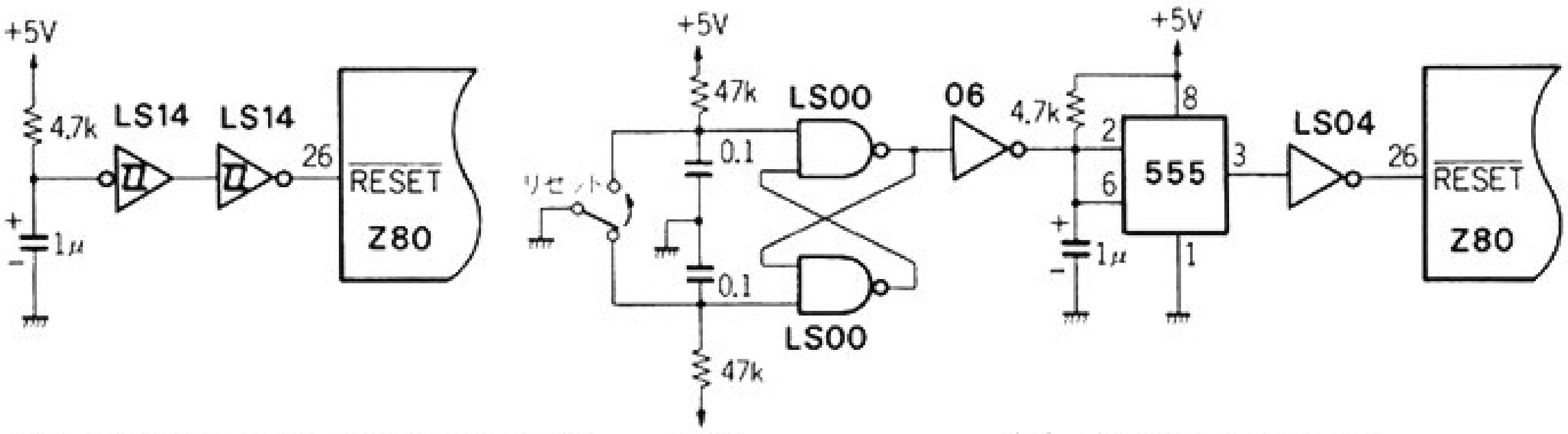
■クロック・ジェネレータ，およびリセット回路

図74にクロック・ジェネレータの例を二つ示します．図74(a)はクリスタルとTTLによるオシレータを用いた場合で，



(a) TTL によるオシレータ　　(b) 市販のオシレータを用いる

〈図 74〉
Z 80 用のクロック・ジェネレータ



(a) LS14 によるパワー・オン・クリヤ回路　　(b) 555 を用いたリセット

〈図 75〉
リセット回路

図(b)は市販の 14 ピンDIPタイプのオシレータを用いた場合です．

Z 80 のクロックの"H"レベルの最小値は，$V_{CC}-0.6$ V と規定されており，したがってふつうは $V_{CC}=5$ V で使用しますので，最低 4.4 V 必要となります．このため Z 80 のクロック入力に接続されているインバータの出力は 330 Ωの抵抗により，プルアップされています．

またクロック入力の立ち上がり/立ち下がり時間に対するスペックも結構きびしいのですが(Z 80，Z 80 A：30 ns，Z 80 B：20 ns)，筆者の経験では図 74 の回路で動作しなかったことは一度もありません．

図 75 にリセット回路の例を二つ示します．図 75(a)は 7414 を用いたパワー・オン・リセット回路です．図(b)はタイマ用IC 555 を用いたリセット回路であり，パワー・オン・リセット，およびマニュアル・リセットができます．

第2章

Z80 CPU命令編

この章では前半でZ80の命令の概念，アドレッシング・モード，フラグ・レジスタの機能について説明をし，後半では個々の命令の意味について詳解し，さらに比較的多用されるであろうと思われる，サブルーチン集を紹介します．

Z80はなぜ命令数が多いのか

8080 Aの命令数が78種であるのに対して，Z80の命令数は158種もあります．Z80は8080 Aが持っている命令をすべて包含し，かつ80種も多くの命令を持っています．ここでいう命令数というのは，“基本命令”の数であり，レジスタによるバリエーションまでを含めていません．

レジスタによるバリエーションまでを含めて考えると，8080 Aの命令数は244種となり，Z80では実は696種となります．ではまはじめにZ80はどのようにして，このように命令数を増やしたのかについて考えてみましょう．

8080 Aの命令のOPコードは1バイトです．図1に8080 Aで使われているOPコードを示します．図中のアミのかかっていない白い部分のコードは，8080 Aでは使われていないコードです．図からわかるように空いているOPコードはたったの12個だけなのです．したがって，ふつうに考えたのでは，わずか12種類の命令しか増やすことはできません．

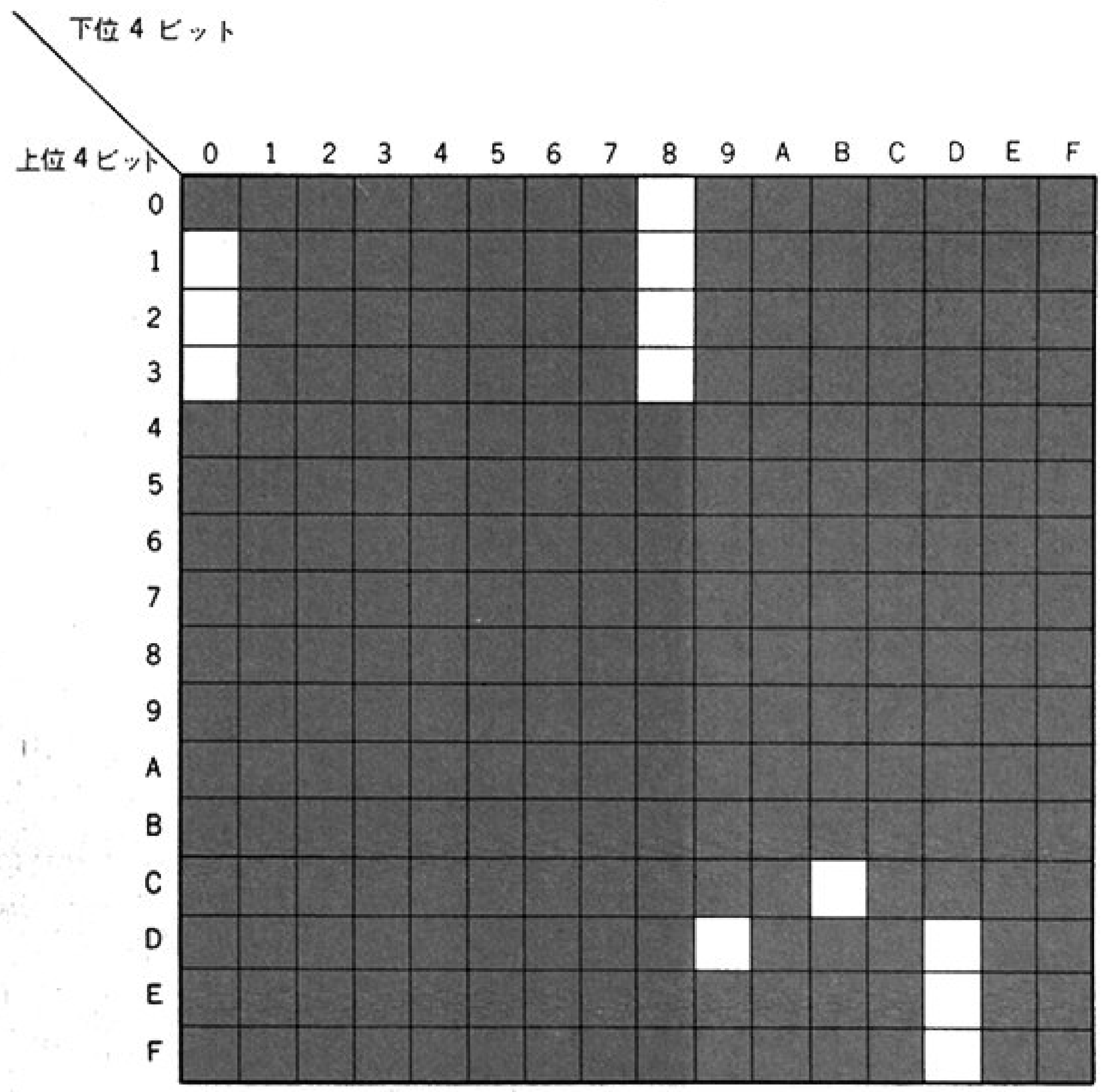
ではZ80ではどのようなくふうをしたのでしょうか?．Z80ではもちろんこれらの空いているOPコードをすべて利用しましたが，その中のOPコード，DDH，EDH，ならびにFDHに関しては“第2OPコード”という概念を導入しました．つまりこれら三つのOPコードを第1OPコードとし，まず8080 Aの命令ではないことを宣言し，その後に続くOPコード(第2OPコード)により命令の種類を規定しました．

このくふうによりDDH，EDH，FDHなどのOPコードに続く，1バイトの第2OPコードが命令の内容を示すのですから，これだけでも256×3=768種の命令を増やすことができます．図2に8080 AのOPコードの空きを，Z80ではどのように使ったかを示します．

第2OPコードを設けたことによる欠点は明らかです．第1OPコードは単にZ80固有の命令であることを示す記号(印)でしかないのですから，命令の長さが1バイト長くなってしまいます．Z80では第2OPコードを持つ命令をフェッチする際には，M1サイクルを2度実行します．したがって命令の長さが1バイト増えてしまうだけではなく，命令の実行時間も4ステート多くかかることになります．

しかしながら，この欠点も第2OPコードを持つ命令

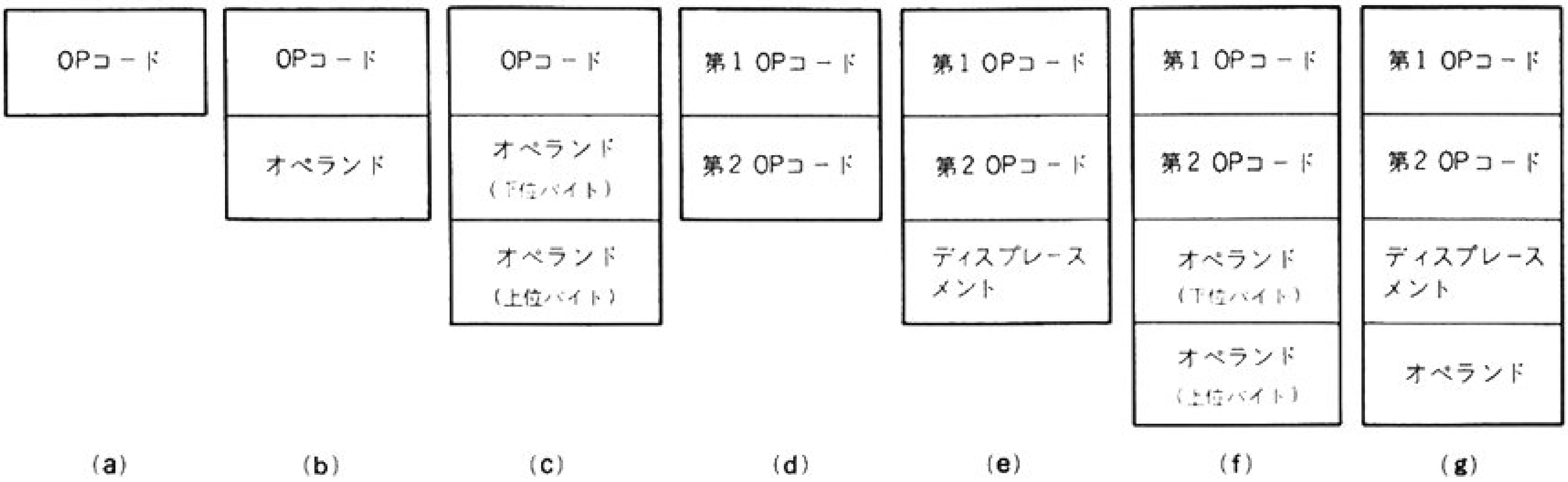
〈図1〉8080 Aで使用されているOPコード(アミの部分，空白部は空きコード)



〈図2〉Z80における8080 Aの空きコードの使いかた

空きOPコード	命　令
0 8	EX AF,AF'
1 0	DJNZ
1 8	JR e
2 0	JR NZ,e
2 8	JR Z,e
3 0	JR NC,e
3 8	JR C,e
C B	ビット操作命令
D 9	EXX
D D	IXを使う命令
E D	その他の命令
F D	IYを使う命令

〈図3〉Z80の命令形式



の機能が十分強力であり，8080 Aの命令をいくつか組み合わせて実行する場合と比べて，命令の長さが短く，実行時間も速ければメリットに変じるわけです．例をあげて説明をしましょう．

いまインデックス・レジスタIXが示す値に，40Hを加えた値が示すメモリの内容をアキュムレータに加えることを考えてみます．Z80ではこれを次の1命令で実行してしまいます．

```
ADD     A, (IX+40H)
```

この命令の長さが3バイト(DDH, 86H, 40H)で実行時間は19ステートです．

これと同じような機能を，8080 Aの命令で実行すると，

```
LXI     B, 40H
DAD     B
ADD     M
```

となり，命令の長さは5バイト，実行時間は27ステートとなります．その上レジスタBCを使っているので，その値を保存する必要がある場合にはさらに命令が増え，実行時間も長くなります．

この例はほんの一例にすぎません．Z80の第2OPコードを持つ命令は非常に強力であり，第2OPコードを持つことが欠点となるような命令は，こと8080 Aと比較する限りにおいてはありません．

Z80の命令の形式

Z80には158種類の命令がありますが，これらの命令はすべてある決まった一定の形式を持っています．図3にZ80の命令形式を示します．

図(a)は1バイト命令です．この種の命令はOPコードだけからなります．図(b)は2バイト命令です．この種の命令は，OPコードと1バイトのオペランドからなります．オペランドはイミーディエイト・データであることもありますし，ジャンプ先に対する相対アドレスであることもあります．

また図(c)は3バイト命令であり，OPコードと2バイトのオペランドからなります．2バイトのオペランドがある場合には，オペランドの下位バイトが先で，上位バイトが後になります．この規則は8080 Aにおいて設定されたものであり，Z80においても互換性の面から踏襲(とうしゅう)されています．2バイトからなるオペランドは，イミーディエイト・データであることもありますし，データやジャンプ先のアドレスを示すこともあります．

8080 Aの命令形式は以上の(a)～(c)のタイプに限られています．Z80において新しく設けられた命令の中には，図(a)および(b)のタイプに含まれるものもありますが，大半は以下に述べる図(d)～(e)の形式を持っています．

図3(d)の形式の命令は2バイト命令ではありますが，両バイトともOPコードです．そして図(e)の命令は，2バイトのOPコードと，1バイトのディスプレースメントからなります．こ種の命令はインデックス・レジスタを用いる命令です．図(f)の命令は2バイトのOPコードと，やはり2バイトのオペランドを持っています．オペランドの下位バイト，上位バイトについての約束ごとは図(c)の場合と同じです．

図(f)の命令はやはり2バイトのOPコードを持っていますが，残る2バイトはディスプレースメントとオペランドです．この場合オペランドはイミーディエイト・データです．

Z80のアドレッシング・モード

アドレッシング・モードというのは，CPUが命令を実行するときに，命令実行の対象となるデータ(オペランド)をどこから持ってくるか，またはどこへしまうかを規定するモードです．

命令の機能を学ぶには，アドレッシング・モードに

〈図4〉イミーディエイト・モード

〔例〕LD A,05H

OPコード	3EH
オペランド（イミーディエイト・データ）	05H

(a)

〔例〕LD (IX+40H),05H

第1 OPコード	DDH
第2 OPコード	36H
ディスプレースメント	40H
オペランド（イミーディエイト・データ）	05H

(b)

〈図5〉拡張イミーディエイト・モード

〔例〕LD HL,1234H

OPコード	21H
オペランド（下位バイト）	34H
オペランド（上位バイト）	12H

(a)

〔例〕LD IX,5678H

第1 OPコード	DDH
第2 OPコード	21H
オペランド（下位バイト）	78H
オペランド（上位バイト）	56H

(b)

関する理解は不可欠です．Z80には10種類のアドレッシング・モードがあります．以下にそれらの個々について説明します．

■イミーディエイト・モード(Immediate)

イミーディエイト・モードは命令自体の中に，その命令が対象とする1バイトのオペランドが含まれている場合のアドレッシング・モードです．図4にイミーディエイト・モードの命令の形式を示します．図(a)はふつうのイミーディエイト・モードの命令です．この種の命令ではOPコードの次のバイトをオペランド(イミーディエイト・データ)として取り扱います．

例としては，

LD　A，05H

命令をあげることができます．この命令のOPコードは3EHで，「OPコードの次のバイトをアキュムレータに入れる」ことを意味します．したがってOPコードの次のバイト05Hがアキュムレータに入れられます．

図(b)の場合は4バイト命令であり，2バイトのOPコードとディスプレースメント，それにオペランドを持っています．例としてあげたのは，

LD　(IX+40H)，05H

命令です．

この命令ではインデックス・レジスタIXにディスプレースメント40Hを加えたアドレスにオペランド(05H)をストアします．この例はイミーディエイト・モードとインデックス・モードを組み合わせたものですが，命令の第4バイトがイミーディエイト・データとして取り扱われますので，まさしくイミーディエイト・モードの命令です．

■拡張イミーディエイト・モード
(Extended Immediate)

このアドレッシング・モードを持つ命令は3バイトまたは4バイトからなります．図5に拡張イミーディエイト・モードの命令の形式を示します．図(a)は3バイト命令であり，OPコードの後に2バイトのオペランドが続きます．この種の命令の例として

LD　HL，1234H

をあげました．この命令のOPコードは21Hであり，「レジスタ・ペアHLに，OPコードに続く2バイトのオペランドを入れる」という意味を持ちます．この命令を実行すると，OPコードの次のオペランドがレジスタLに入れられ，その次のオペランドはレジスタHに入れられます．

図(b)は4バイト命令の場合です．この種の命令としては，インデックス・レジスタに対してイミーディエイト・データをストアする場合をあげることができます．この例では，

LD　IX，5678H

を取りあげていますが，この命令の意味は「オペランドの下位バイトをIXの下位バイトに，オペランドの上位バイトのIXの上位バイトに入れる」ことです．

したがって，この例の命令を実行すると，オペランドの下位バイト(78H)がIXの下位8ビットにストアされ，オペランドの上位バイトがIXの上位8ビットにストアされます．

■ページ・ゼロ・モード(Page Zero)

このアドレッシング・モードを持つ命令は，RST命令だけです．図6にRST命令の形式を示します．RST命令は1バイト命令であり，命令のビット3～5の値をnとすると，8×n番地に対するCALL命令を実行します．

したがって，nの値がゼロのときには0000H番地に対してCALL命令を実行し，nの値が7のときには38H(7×8=56)番地に対してCALL命令を実行します．

ページ・ゼロという意味は，アドレスの上位8ビットが常にゼロであることを意味します．ページという

概念は必須のものではありませんが，256バイトを1ページと呼ぶことがあり，0000H～00FFH番地のことをページ・ゼロと呼んだりします．

■相対アドレス・モード(Relative)

図7に相対アドレス・モードの命令の形式を示します．Z80において相対アドレス・モードが使われるのは，相対ジャンプ命令(JRおよびDJNZ)だけです．相対ジャンプ命令は2バイトからなり第1バイトはOPコード，第2バイトはオペランドであり，このオペランドはジャンプ先に対する相対アドレスです．

図7の例は2000H番地にあるJR命令であり，そのオペランドは46Hです．この命令を実行するとCPUはJR命令の次の番地(2002H)にオペランドの符号ビットを拡張した16ビットの値(0046H)を加えた値を求め(2002H＋0046H＝2048H)，その値が示すアドレス2048H番地へジャンプします．つまり相対アドレス・モードでは，PCの値＋オペランドの値(符号拡張)により，実行命令のアドレスが求められます．

今の例はオペランドの値がプラスでしたが，オペランドのビット7(符号ビット)が"1"であると，マイナスになります．図7の例のオペランドの値を80H(－128)とした場合を考えてみましょう．この場合にはオペランドの符号ビットが左に拡張され，16ビットの値FF80Hが作られ，PCの値に加えられます．したがって，

2002H＋FF80H＝1F82H

の計算が行われ，1F82H番地にジャンプすることになります．この計算の過程において，ビット15からビット16へのキャリーは無視されます．

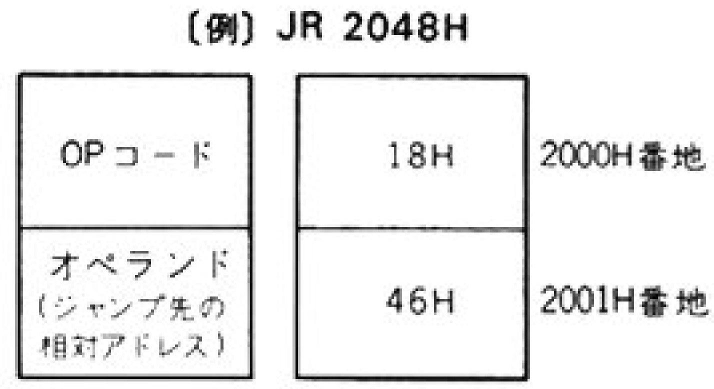
相対アドレス・モードにおけるオペランドのとり得る値は00H～FFHで，00H～7FHがプラス，80H～FFHがマイナスです．したがってPCが示す値の－128(80H)～＋127(7FH)の範囲をアドレッシングすることができます．

〈図6〉ページ・ゼロ・モード(RST命令)

7	6	5	4	3	2	1	0
1	1	×	×	×	1	1	1

5	4	3	飛び先
0	0	0	0000H
0	0	1	0008H
0	1	0	0010H
0	1	1	0018H
1	0	0	0020H
1	0	1	0028H
1	1	0	0030H
1	1	1	0038H

〈図7〉
相対アドレス・モード



■拡張アドレス・モード(Extended)

拡張アドレス・モードを持つ命令は，3バイトまたは4バイトからなり，1バイトまたは2バイトのOPコードの後に2バイトからなるオペランド・アドレスが続きます．図8に拡張アドレス・モードの命令の形式を示します．

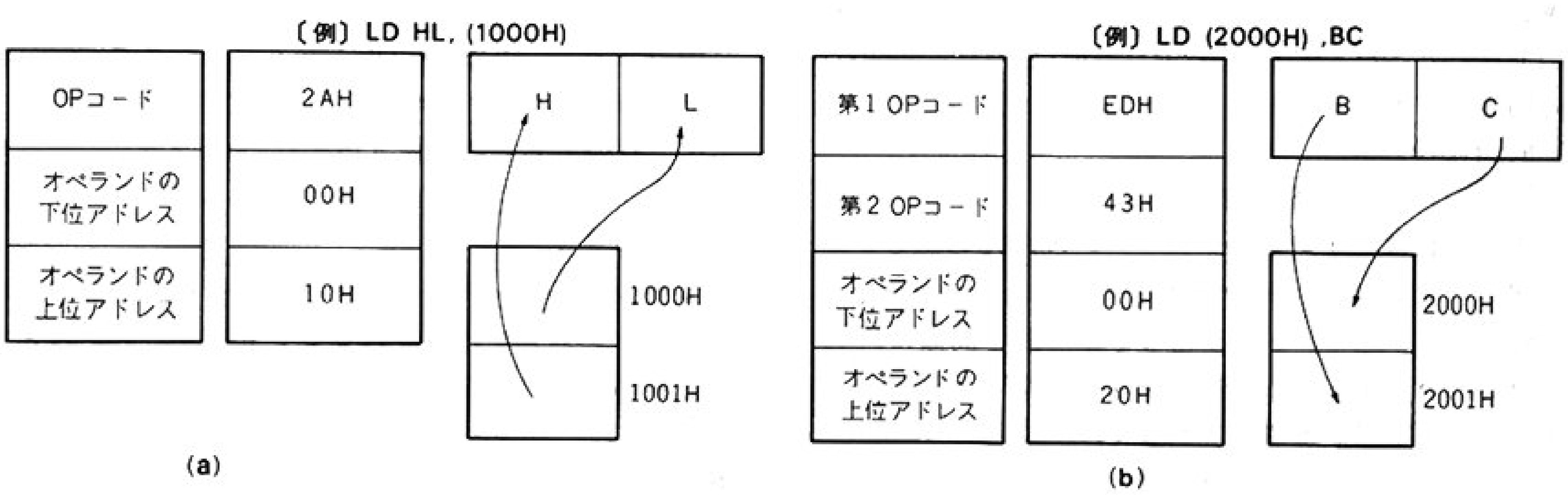
このアドレス・モードの命令はオペランドそのものではなく，オペランドのアドレスを持っています．

図(a)の例は

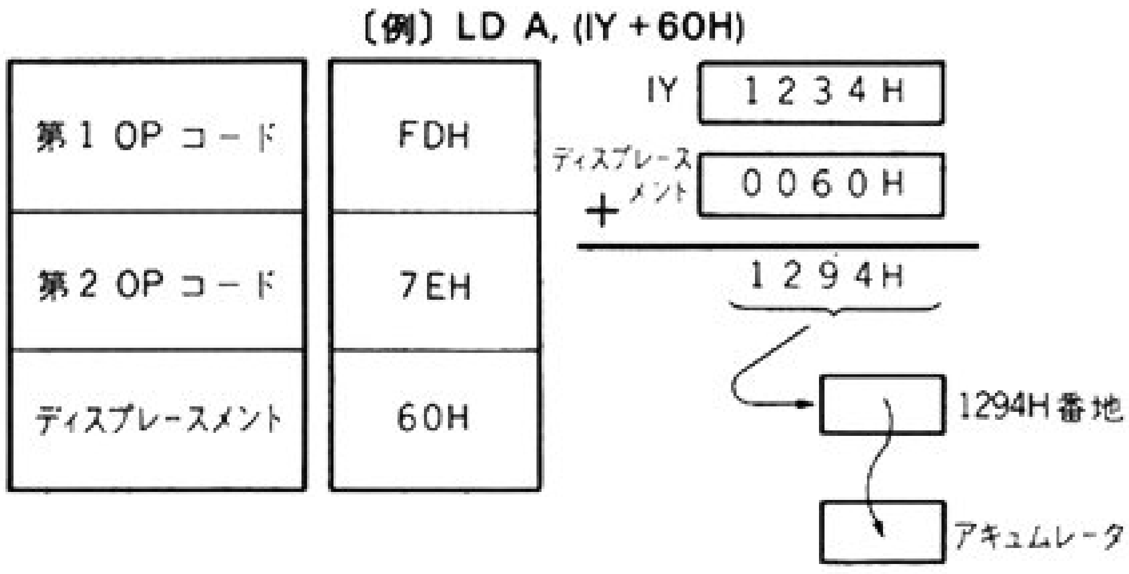
LD　　HL，(1000H)

です．この命令のOPコードは2AHであり，オペランド・アドレスは1000Hです．この命令を実行すると，1000H番地の内容がレジスタLに入れられ，1001H番地の内容がレジスタHに入れられます．

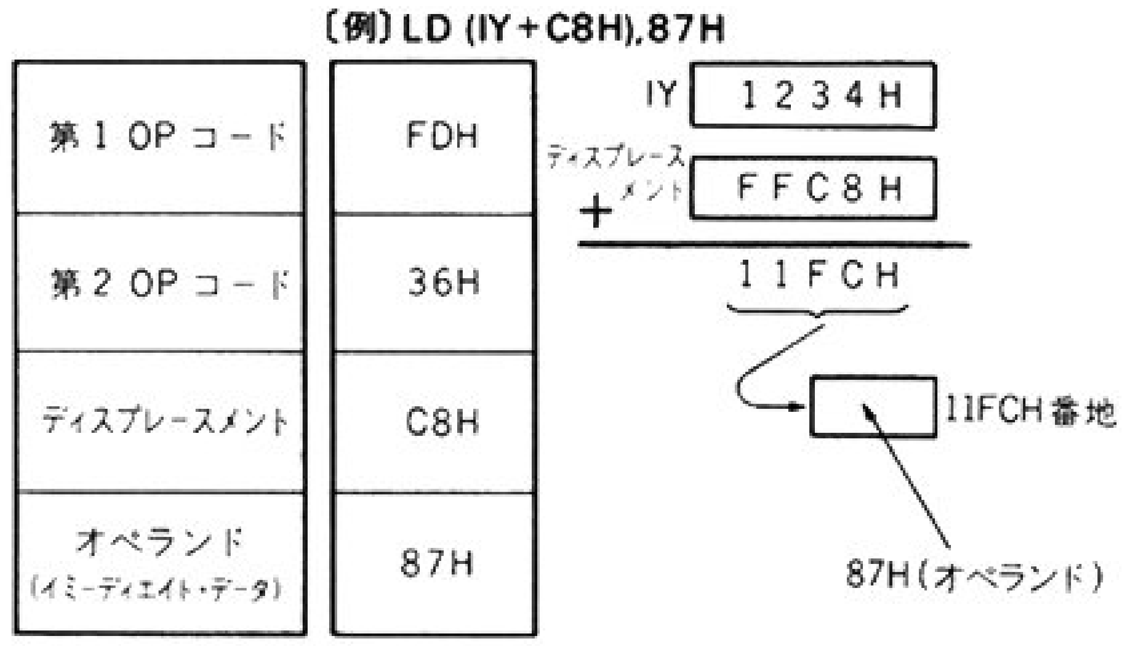
〈図8〉拡張アドレス・モード



〈図9〉インデックス・モード



(a)



(b)

同図(b)の例は，

LD　(2000H)，BC

です．この命令は2バイトのOPコード(EDH，43H)を持ち，その後に2バイトのオペランド・アドレスが続きます．この命令を実行すると，レジスタCの値が2000H番地にストアされ，レジスタBの値が2001H番地にストアされます．

■インデックス・モード(Index)

インデックス・モードは，インデックス・レジスタIX，またはIYを用いる命令において使われます．図9にインデックス・モードの命令の形式を示します．このモードの命令は必ずOPコードが2バイトあり，第1 OPコードは，インデックス・レジスタIXを用いるものは，DDHであり，IYを用いるものはFDHです．

インデックス・モードではオペランドのアドレスは，インデックス・レジスタの値とディスプレースメントの和により決まります．ディスプレースメントは，相対アドレス・モードの場合と同じように，符号拡張されます．

図9(a)の例は，

LD　A，(IY＋60H)

です．この命令は「インデックス・レジスタIYとディスプレースメント60Hを加えたアドレスの内容を，アキュムレータに入れる」ことを意味します．IYの値が1234Hであったとすると，1294H番地(1234H＋0060H)の内容がアキュムレータに入れられます．

同図(b)の例は，

LD　(IY＋C8H)，87H

です．これは「インデックス・レジスタIYの値とディスプレースメントC8Hを加えたアドレスに，オペランド87Hを書き込め」という命令です．前と同様にIYの値を1234Hとすると，11FCH番地(1234H＋FFC8H)にオペランド87Hが書き込まれます．

インデックス・レジスタの値とディスプレースメントの値を加える場合，ディスプレースメントの符号ビット(ビット7)は左に拡張され，16ビットの値としてインデックス・レジスタの値と加えられます．

ディスプレースメントのとり得る値は00H～FFHなので，インデックス・モードの場合にも，インデックス・レジスタの値の－128～＋127番地の範囲をアドレッシングすることができます．

■レジスタ・モード(Register)

Z80の命令の多くは，OPコードの中に演算の対象とすべきレジスタを指定するビットを持っています．OPコードの中のビットの組み合せにより，レジスタを指定することをレジスタ・モードのアドレッシングと呼びます．

レジスタ・モードの代表的な例として，

LD　r，r′

命令と，

PUSH　qq

命令を取りあげます．図10にこれらの命令の形式を示します．図10(a)はLD　r，r′命令です．この命令はレジスタr′の内容を，レジスタrに移します．たとえばr′＝000，r＝111の場合にはLD　A，B　となり，レジスタBの内容がアキュムレータに移されます．

同図(b)はPUSH　qq命令です．この命令はレジスタqqの内容をスタックに書き込みます．例としてqq＝01とすると，PUSH　DEとなり，レジスタ・ペアDEの内容がスタック・ポインタが示す値マイナス1(SP－1)のアドレスに書き込まれ，レジスタEの内容がスタック・ポインタが示す値マイナス2(SP－2)に書き込まれます．

このようにレジスタ・モードのアドレッシングでは，単一のレジスタが指定されることもあり，レジスタ・ペアが指定されることもあります．

■インプライド・モード(Implied)

〈図10〉 レジスタ・モード

7	6	5	4	3	2	1	0
0	1	r			r'		

r, r'	レジスタ
0 0 0	B
0 0 1	C
0 1 0	D
0 1 1	E
1 0 0	H
1 0 1	L
1 1 1	A

(a) LD r, r'

7	6	5	4	3	2	1	0
1	1	q	q	0	1	0	1

q q	レジスタ
0 0	B C
0 1	D E
1 0	H L
1 1	A F

(b) PUSH qq

〈図11〉 ビット・モード(BIT b, r)

	7	6	5	4	3	2	1	0
第1 OPコード	1	1	0	0	1	0	1	1
第2 OPコード	0	1	b			r		

b	ビット番号
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

英語のImplyという言葉を辞書で引くと「(暗に)～の意味を含む」とあります．インプライド・モードのアドレッシングでは，命令自体にはレジスタを指定するようなビットはありませんが，命令の機能の約束ごととして演算の対象が決まっています．

Z 80の論理演算命令(ADD, SUB, AND, OR, XOR, CP)では，ソース・オペランドは指定できますが，演算結果が現われるデスティネーションは，常にアキュムレータであると決まっています．このようなアドレッシングをインプライド・モードと呼びます．

■間接アドレッシング(Indirect)

Z 80における間接アドレッシングは，レジスタ・ペアによりメモリ・アドレスを指定することをいいます．このモードの代表的な例は，

LD A, (HL)

です．この命令は，OPコードだけのたった1バイトの命令ですが，比較的大きなことをします．つまり，レジスタ・ペアHLで示されるメモリの内容を，アキュムレータにストアします．この種の命令としてほかに，LD A, (BC)やLD A, (DE)命令があります．

またブロック転送命令(LDIRなど)やブロック・サーチ命令(CPIRなど)も間接アドレッシングを用います．LDIR命令ではソースのアドレスをレジスタ・ペアHLで示し，デスティネーションのアドレスはDEで示されます．

Z 80で新しく設けられた入出力命令，OUT (C), rやIN r, (C)なども間接アドレッシングです．これらの命令の場合，アドレッシングされるのはメモリではなくI/Oですが，レジスタCによりI/Oアドレスが指定されます．

■ビット・モード(Bit)

ビット・モードのアドレッシングは，ビット操作命令(BIT, SET, RES)において用いられます．このモードは，前述の各モードの一段下にあるということができます．

その意味は前述の各モードにしたがって，メモリ，またはレジスタのアドレッシングが決まった後に，はじめてこのビット・モードが有効となるからです．

図11にBIT b, r命令の形式を示します．この命令はレジスタ・モードのアドレッシングを持ち，これによりレジスタrが決定されます．その後第2 OPコード中の3ビットにより操作すべきビットが決まります．

Z 80のフラグ・レジスタ

CPUは単に演算を実行するだけでなく，演算の後に演算結果を調べることができるようになっていなければなりません．CPUは演算をすると同時に，判断のできる機械だからです．そのためには，演算結果をしまっておく何らかの場所が必要です．この場所がフラグ・レジスタです．

Z 80には6ビットのフラグ・レジスタがあります．その中の4ビットは条件付きジャンプ命令により判定することができ，2ビットは10進演算用の制御ビットです．

図12にZ 80のフラグ・レジスタの構成を示します．フラグ・レジスタの各ビットのうち，C, Z, SおよびP/Vは条件付きジャンプ命令により，それらが"1"であるか"0"であるかをテストすることができます．

以下，各フラグ・ビットの意味を説明します．

〈図12〉 Z 80のフラグ・レジスタ

	7	6	5	4	3	2	1	0
F	S	Z	X	H	X	P/V	N	C

S : Sign (符号)
Z : Zero (ゼロ)
H : Half Carry (ハーフ・キャリー)
P/V : Parity/Overflow (パリティ/オーバフロー)
N : Subtract Flag (減算フラグ)
C : Carry (キャリー)
X : 不使用

■キャリー・フラグ(C)

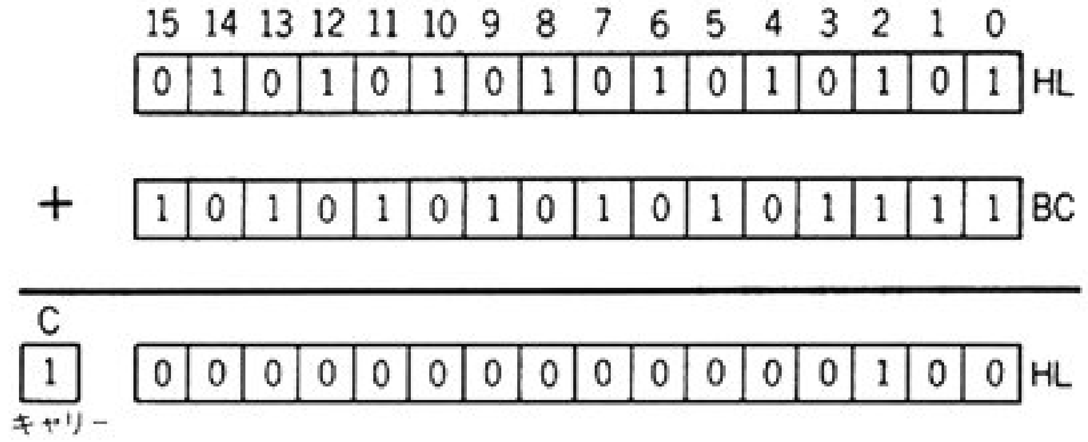
加算命令(ADD, ADC)の場合には，演算結果の最上位ビットから桁上がりを生じたときにセット(“1”)されます．減算命令(SUB, SBC, NEG, CP)の場合には，上位桁から借り(ボロー)を生じたときにセットされます．

また，データをシフト，または回転するローテイト命令では最上位ビット(左シフトのとき)，または最下位ビット(右シフトのとき)の状態がキャリー・フラグにセットされます．

キャリー・フラグを直接コントロールするSCF命令を実行すると，キャリー・フラグは強制的に“1”にセットされ，CCF命令を実行するとキャリー・フラグの状態は反転(0 → 1, 1 → 0)します．論理演算命令(AND, OR, XOR)を実行すると，キャリー・フラグは必ず“0”になります．

図13にADD A, B命令を実行したときのキャリー・フラグのようすを示します．図13(a)では桁上がりが発生していないので，キャリー・フラグは“0”となり，図(b)では桁上がりが生じているので，キャリー・フラグは“1”になっています．

〈図14〉 ADD HL, BC命令を実行したときのキャリー・フラグ(C)



HL=5555 H, BC=AAAFHのとき，実行結果はHL=0004 H, C=1となる

図14にはレジスタ・ペアに対する加算命令，ADD HL, BC を実行したときのキャリー・フラグのようすを示します．この例では演算結果(HL)のビット15からの桁上がりが発生しているため，キャリー・フラグは“1”になっています．

図15には減算命令，SUB B を実行したときのキャリー・フラグのようすを示します．減算命令は，引く数の補数を加えることにより実行され，加算時のキャリーを反転したものがキャリー・フラグにセットされます．

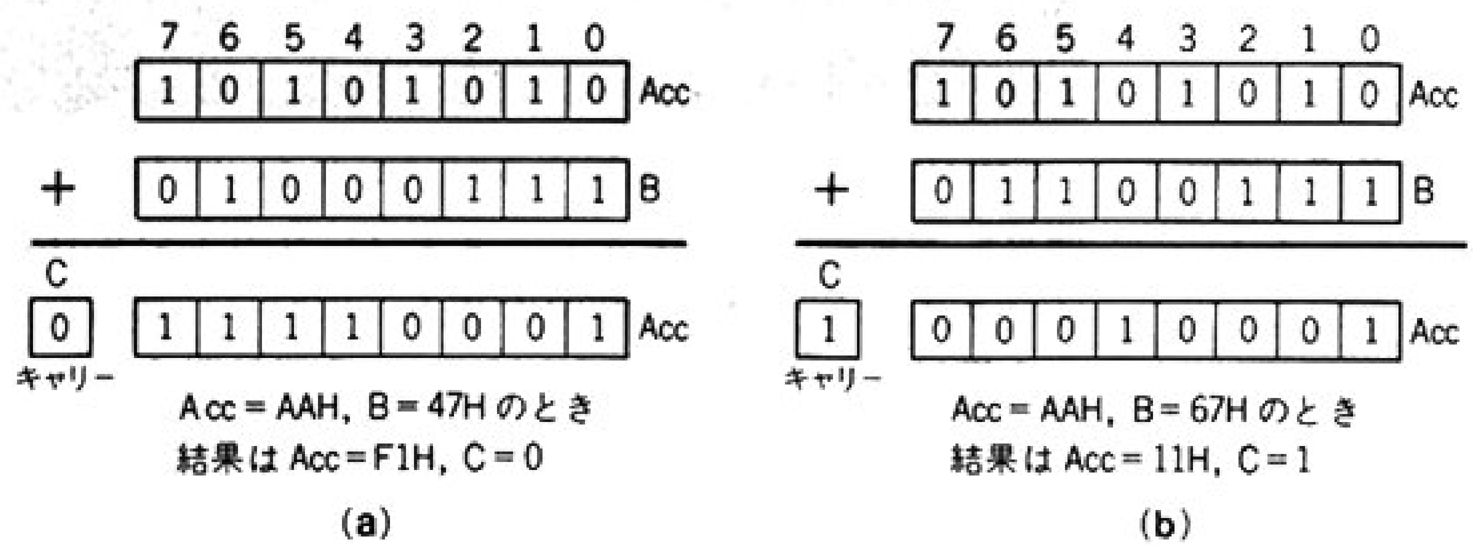
図(a)では補数を加算したときにキャリーが発生しますので，キャリー・フラグは“0”(桁借りなし)となり，図(b)では補数の加算時にキャリーが生じないので，キャリー・フラグは“1”(桁借りあり)となります．

■ゼロ・フラグ(Z)

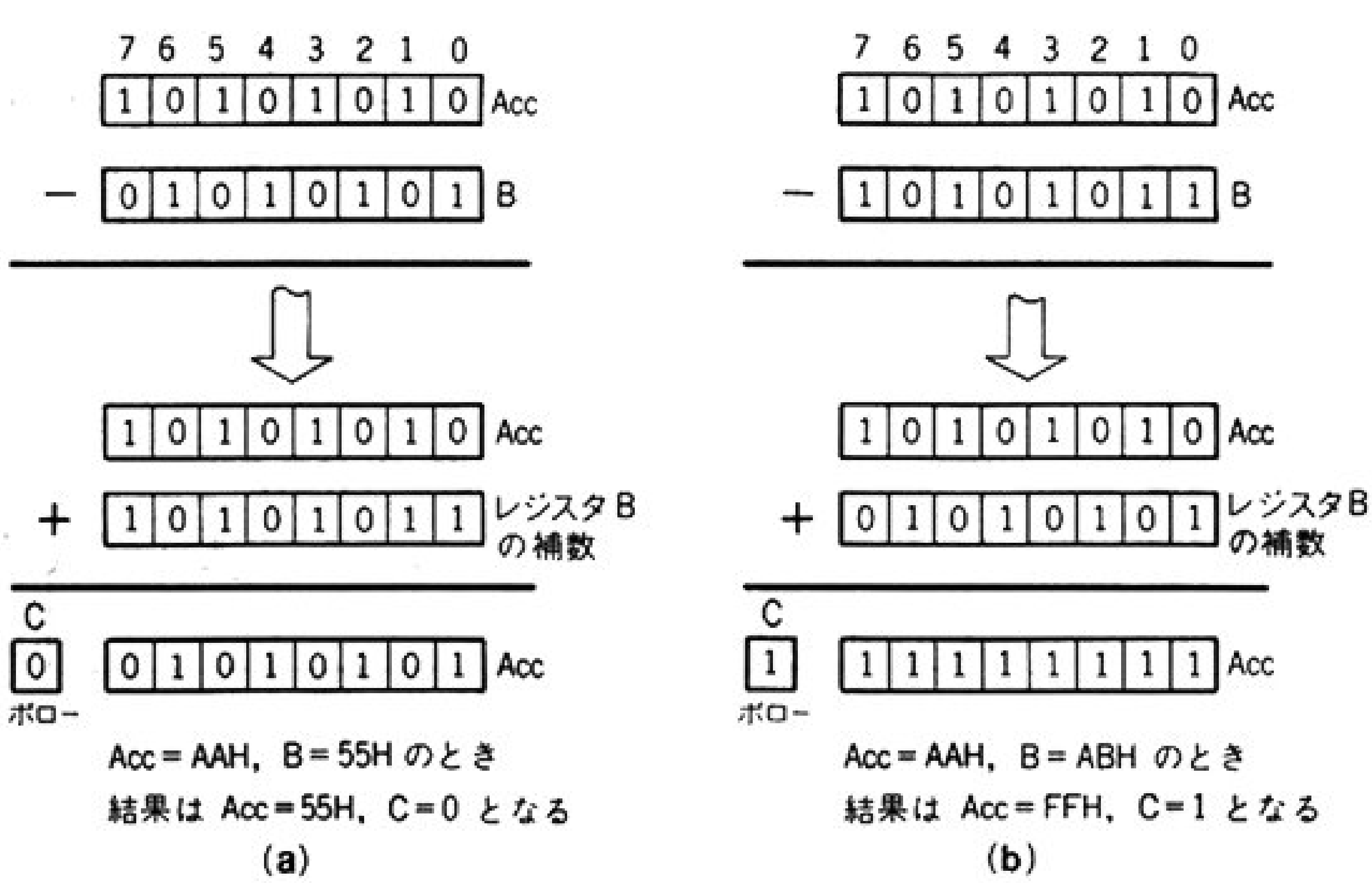
ゼロ・フラグはその名のとおり，演算結果がゼロであることを示すフラグです．加算命令(ADD, ADC)，減算命令(SUB, SBC, CP, NEG)，論理演算命令(AND, OR, XOR)，インクリメント/デクリメント命令(INC, DECならびにDAA)において，演算結果がゼロであるときに，このゼロ・フラグは“1”になり，ゼロでないときには“0”になります．

またZ80で新設されたローテイト/シフト命令(RL, RLC, RR, RRC, SLA, SRA, SRL, RLD, RRD)では，ゼロ・フラグが変化します．さらに8080AコンパチブルなIN A, (n)命令ではゼロ・フラグは変化しませんが，Z80で登場したIN r, (C)命令では読み込まれたデータにしたがって，ゼロ・フラグは変化します．

〈図13〉 ADD A, B命令を実行したときのキャリー・フラグ(C)



〈図15〉 SUB B命令を実行したときのキャリー・フラグ(C)



〈図 16〉
符号付き整数の表現

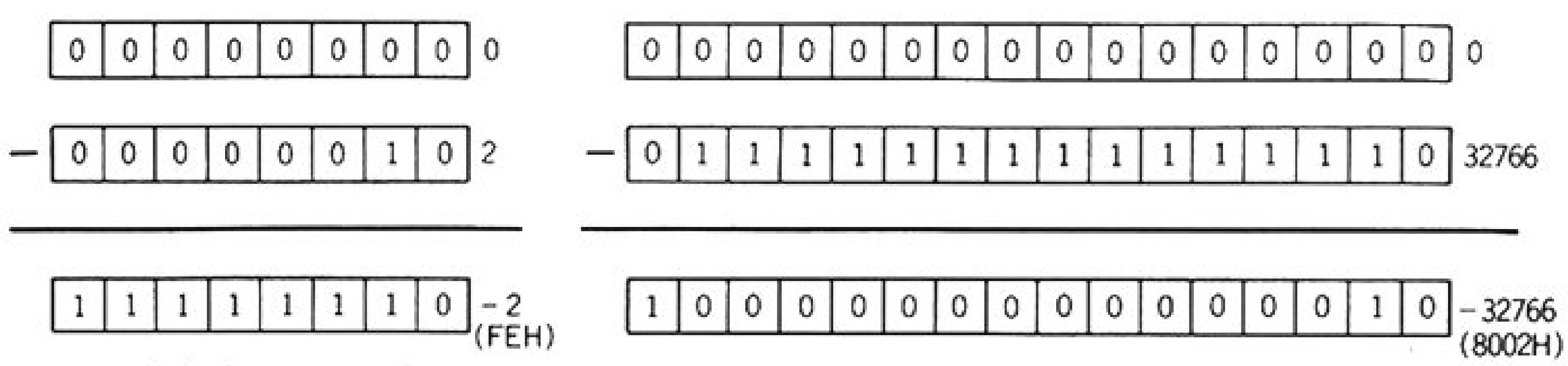
値	16進数	ビット・パターン
127	7F	0 1 1 1 1 1 1 1
126	7E	0 1 1 1 1 1 1 0
2	02	0 0 0 0 0 0 1 0
1	01	0 0 0 0 0 0 0 1
0	00	0 0 0 0 0 0 0 0
−1	FF	1 1 1 1 1 1 1 1
−2	FE	1 1 1 1 1 1 1 0
−127	81	1 0 0 0 0 0 0 1
−128	80	1 0 0 0 0 0 0 0

8ビット

値	16進数	ビット・パターン
32767	7FFF	0111111111111111
32766	7FFE	0111111111111110
2	0002	0000000000000010
1	0001	0000000000000001
0	0000	0000000000000000
−1	FFFF	1111111111111111
−2	FFFE	1111111111111110
−32767	8001	1000000000000001
−32768	8000	1000000000000000

16ビット

〈図 17〉
負の数を求める計算



(a) 8ビットの−2　　(b) 16ビットの−32766

ブロック入/出力命令(INI, IND, OUTI, OUTD, INIR, INDR, OTIR, OTDR)では，ゼロ・フラグはレジスタBの値がゼロであるか否かを示します．したがって，これらの命令のうち，×××R(Repeat)という命令はレジスタBの値がゼロになるまで反復されるので，この種の命令の実行終了時のゼロ・フラグは必ず“1”になります．

ストリング・サーチ命令(CPI, CPIR, CPD, CPDR)では，ストリング中に目的のデータが見つかったときに，ゼロ・フラグは“1”になります．インタラプト・レジスタやリフレッシュ・レジスタの内容を読む命令(LD A, IやLD A, R)においても，ゼロ・フラグが変化します．

ビット・テスト命令(BIT b, s)では，データのテストすべきビットを反転したものがゼロ・フラグにセットされます．したがって，データのテストすべきビットが“1”であるときにはゼロ・フラグは“0”になり，テストすべきビットである場合にはゼロ・フラグは“1”になります．

ゼロ・フラグに関して注意すべきことは，アキュムレータの値を反転するCPL命令では，ゼロ・フラグが変化しないことです．

■サイン・フラグ(S)

サイン・フラグは演算結果の符号を示します．演算結果が正(プラス)であれば，サイン・フラグは“0”になり，負(マイナス)であればサイン・フラグは“1”になります．

サイン・フラグの意味を理解するには，まず符号付きの数の表現を知る必要があります．図16に8ビットおよび16ビットの符号付き整数の表現方法を示します．

符号付き整数の表現では，最上位ビットは符号ビットとして取り扱われ，正の場合には“0”であり，負の場合には“1”となります．8ビット・データの場合には−128(80H)から+127(7FH)の範囲を表現でき，16ビット・データの場合には，−32768(8000H)から+32767(7FFFH)の範囲を表現できます．負の数の表現は，ゼロからその数を引き算すれば求まります．図17に8ビット・データの−2，および16ビット・データの−32766を求める計算を示します．

さて，以上の予備知識を基にして，サイン・フラグの説明をします．サイン・フラグは基本的には加算命令(ADD, ADC)および減算命令(SUB, SBC, CP, NEG)の結果の符号ビットの状態を示していますが，このほかにも論理演算命令(AND, OR, XOR)やINC, DEC命令，ローテイト/シフト命令，IN r, (C)

命令, LD　A, I命令, LD　A, R命令においても, 演算結果の符号ビットの状態がサイン・フラグにセットされます.

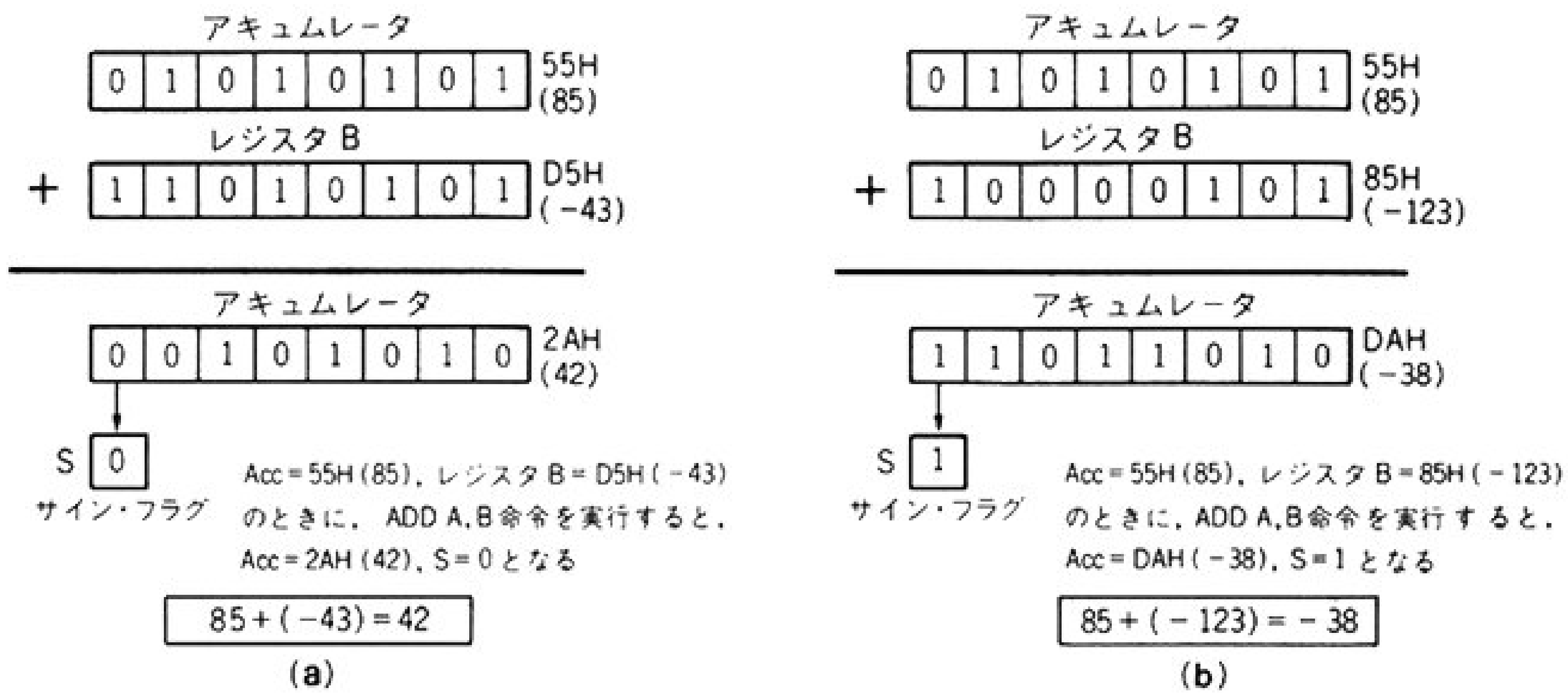
加算命令と減算命令におけるサイン・フラグの動きを例によって見ていきます. 図18にADD命令の場合のサイン・フラグの動きを示します. 図(a)の場合には, 演算結果の符号はプラスなので, サイン・フラグは“0”になっていますが, 図(b)の場合には, 演算結果の符号はマイナスなので, サイン・フラグは“1”になっています.

SUB命令の場合のサイン・フラグの動きを図19に示します. 図(a)の場合には, 演算結果の符号はプラスなので, サイン・フラグは“0”で, 図(b)では演算結果の符号はマイナスなので, サイン・フラグは“1”となります.

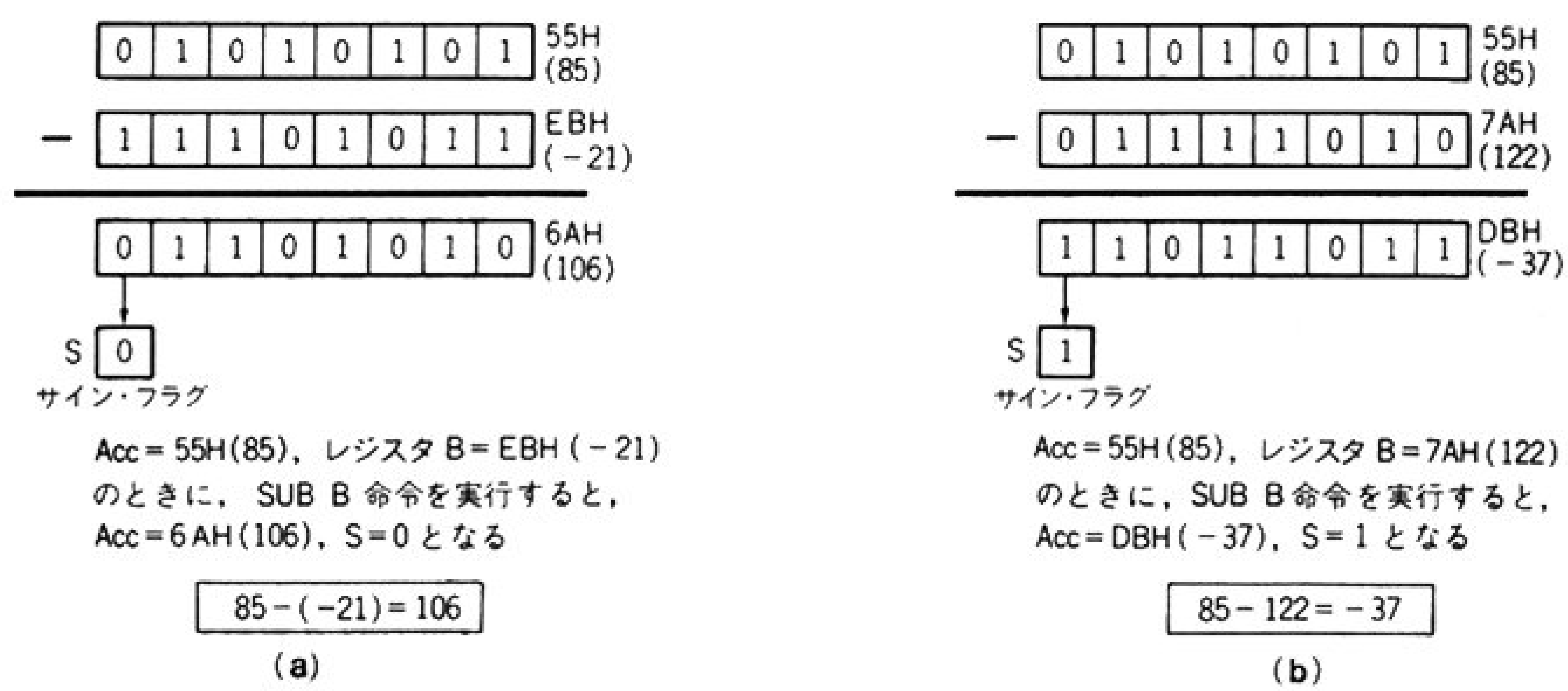
■パリティ/オーバフロー・フラグ(P/V)

このP/Vフラグは大まかにいうと, 演算結果のパリ

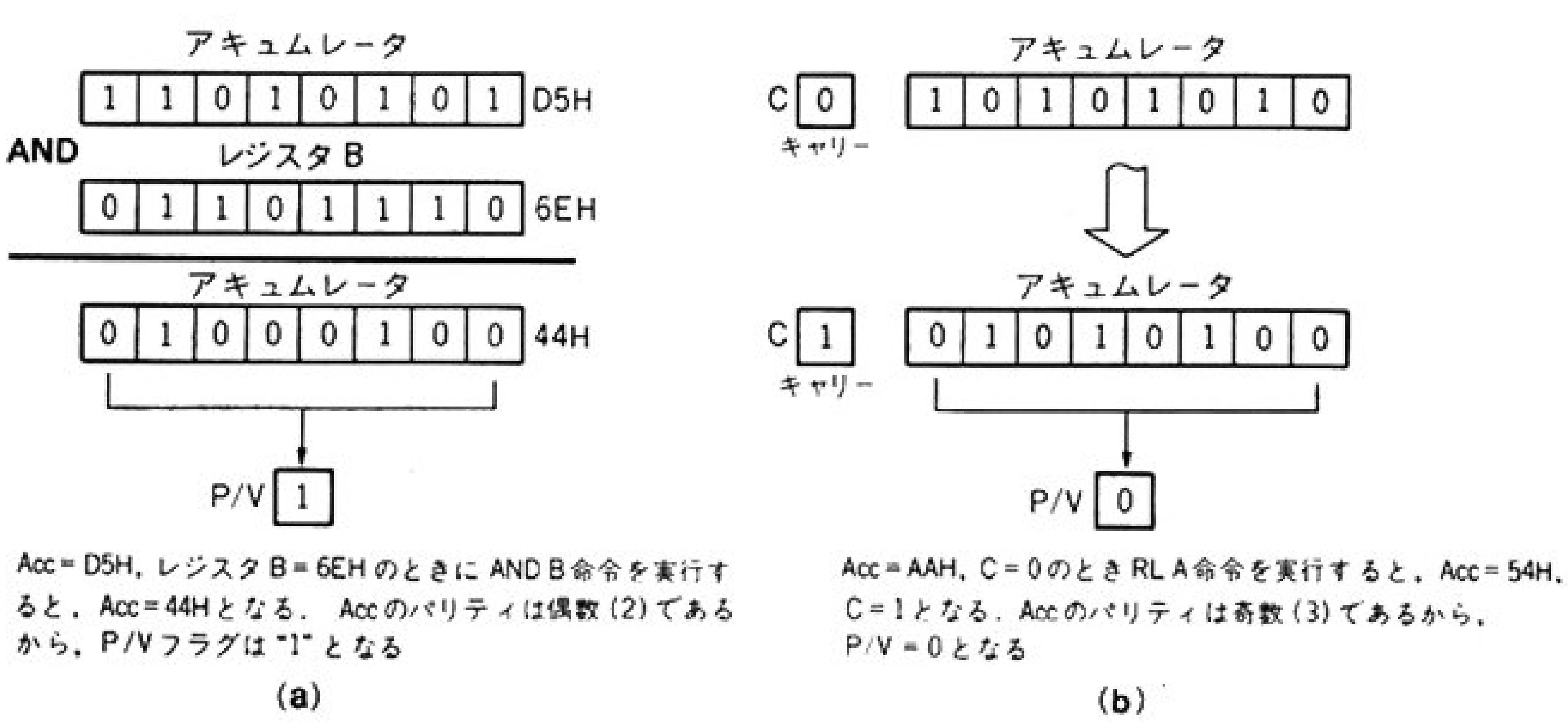
〈図18〉
ADD命令におけるサイン・フラグ



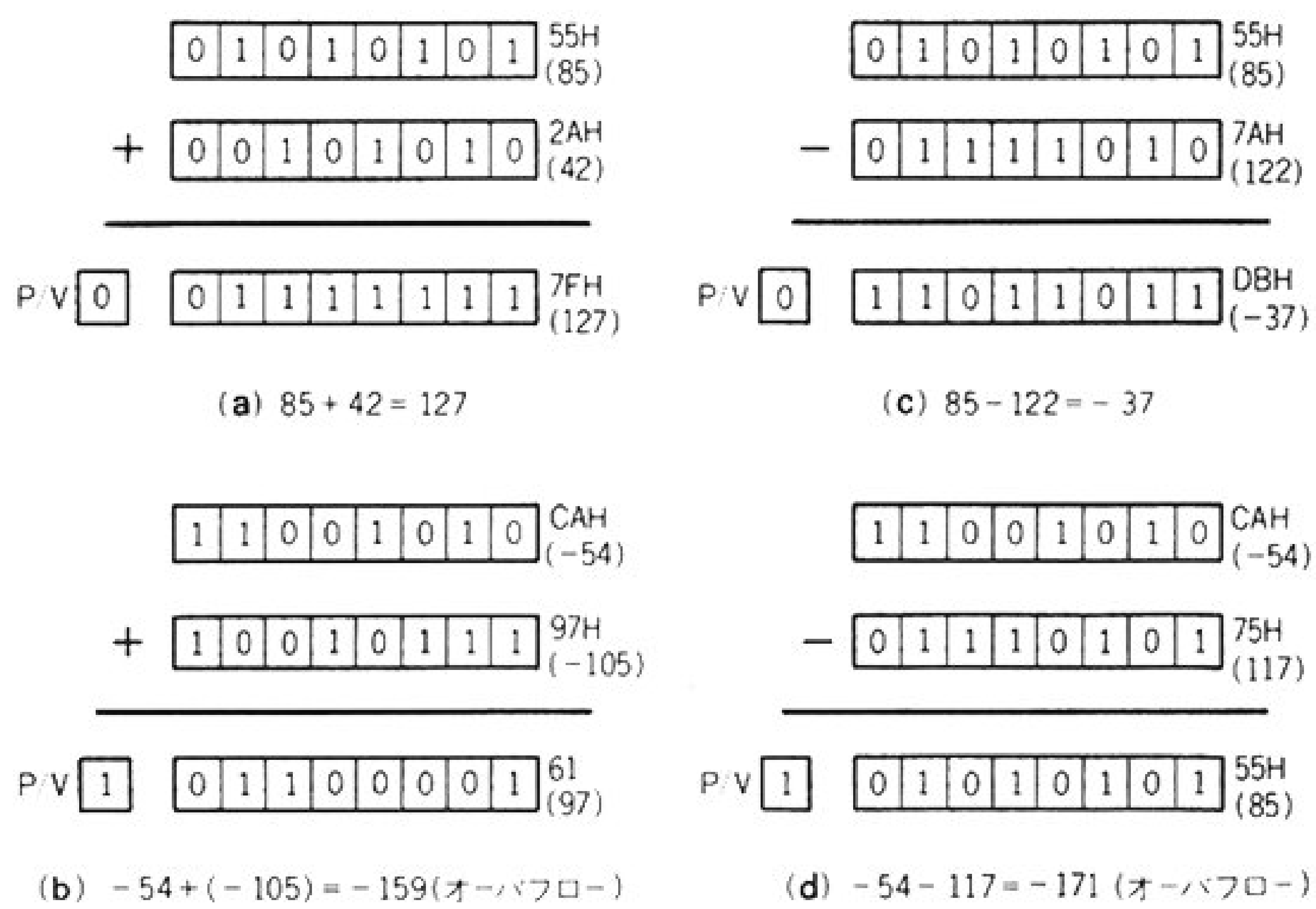
〈図19〉
SUB命令におけるサイン・フラグ



〈図20〉
AND命令, およびRL命令におけるP/Vフラグの動き



〈図21〉符号付き加減算におけるP/Vフラグの動き



ティ極性(偶数のとき"1")，またはオーバフローを示しますが，命令によっては特殊な意味を持つことがあります．

論理演算命令(AND，OR，XOR)やローテイト/シフト命令，それにDAA命令とIN r，(C)命令においては，このP/Vフラグはパリティ極性を示し，演算結果のパリティ("1"の数)が偶数の場合には"1"となり，奇数の場合には"0"となります．図20にAND命令，およびRL命令の場合のP/Vフラグの動きを示します．図(a)では演算結果のパリティが偶数なので，P/Vフラグは"1"，図(b)では演算結果のパリティが奇数なので，P/Vフラグはゼロになります．

加算命令(ADD，ADC)や減算命令(SUB，SBC)において，符号付きの演算を行った場合，結果にオーバフローが生じたとき，このP/Vフラグは"1"になります．オーバフローというのは，演算結果の数がその演算が対象とするデータの幅を越えなければ，表現できないような状態をいいます．

符号付きの表現では8ビット・データで表わすことのできる数の範囲は−128(80H)〜＋127(7FH)です．したがって，8ビットの加算，または減算を行い，結果の値が前記の範囲を越えた場合には，オーバフローとなります．16ビット演算の場合には，演算結果が−32768(8000H)〜＋32767(7FFFH)の範囲を越えた場合，オーバフローとなります．

いいかたを換えますと，オーバフロー・フラグがセットされるのは，両オペランドの符号ビットがともに正("0")で，結果の符号ビットが負("1")である場合と，両オペランドの符号ビットがともに負で，結果の符号ビットが正("0")である場合です．二つのオペランドの符号ビットをAnおよびBnとし，結果の符号ビットをRnとした場合，P/Vフラグの状態を示す論理式は，

$$P/V = Rn \cdot \overline{An} \cdot \overline{Bn} + \overline{Rn} \cdot An \cdot Bn$$

となります．

図21に8ビットの加減算におけるP/Vフラグの動きを示します．図(a)と(c)では，演算結果は−128〜＋127の範囲に入っているので，P/Vフラグは"0"で，図(b)と(d)では，演算結果は−128〜＋127の範囲を越えるため，P/Vフラグは"1"になっています．

このほかにP/VフラグはINC命令，およびDEC命令においても変化します．INC命令では演算前のオペランドの値が7FHのときにP/Vフラグはセットされ，DEC命令の場合には，演算前のオペランドの値が80 Hのときにセットされます．

INCおよびDECという命令を，それぞれ「1を加算する命令」，および「1を減算する命令」ととらえれば，このP/Vフラグの動きを理解することができます．

またこのP/Vフラグは，ブロック移動命令(LDI，LDD)やストリング・サーチ命令(CPI，CPD，CPIR，CPDR)においては，レジスタ・ペアBCの値がゼロであるか否かを示します．レジスタ・ペアBCの値がゼロであるときは，P/Vフラグは"0"になり，ゼロでないときには，P/Vフラグは"1"になります．

さらにこのP/Vフラグが特殊な意味に使われているのは，LD A，I命令とLD A，R命令です．これらの命令では，P/VフラグはZ80内部の割り込みイネーブル・フリップフロップ(IFF)の状態を示します．

P/VフラグはIFFがセットされていれば(割り込み可)"1"となり，IFFがリセットされていると(割り込み不可)"0"になります．

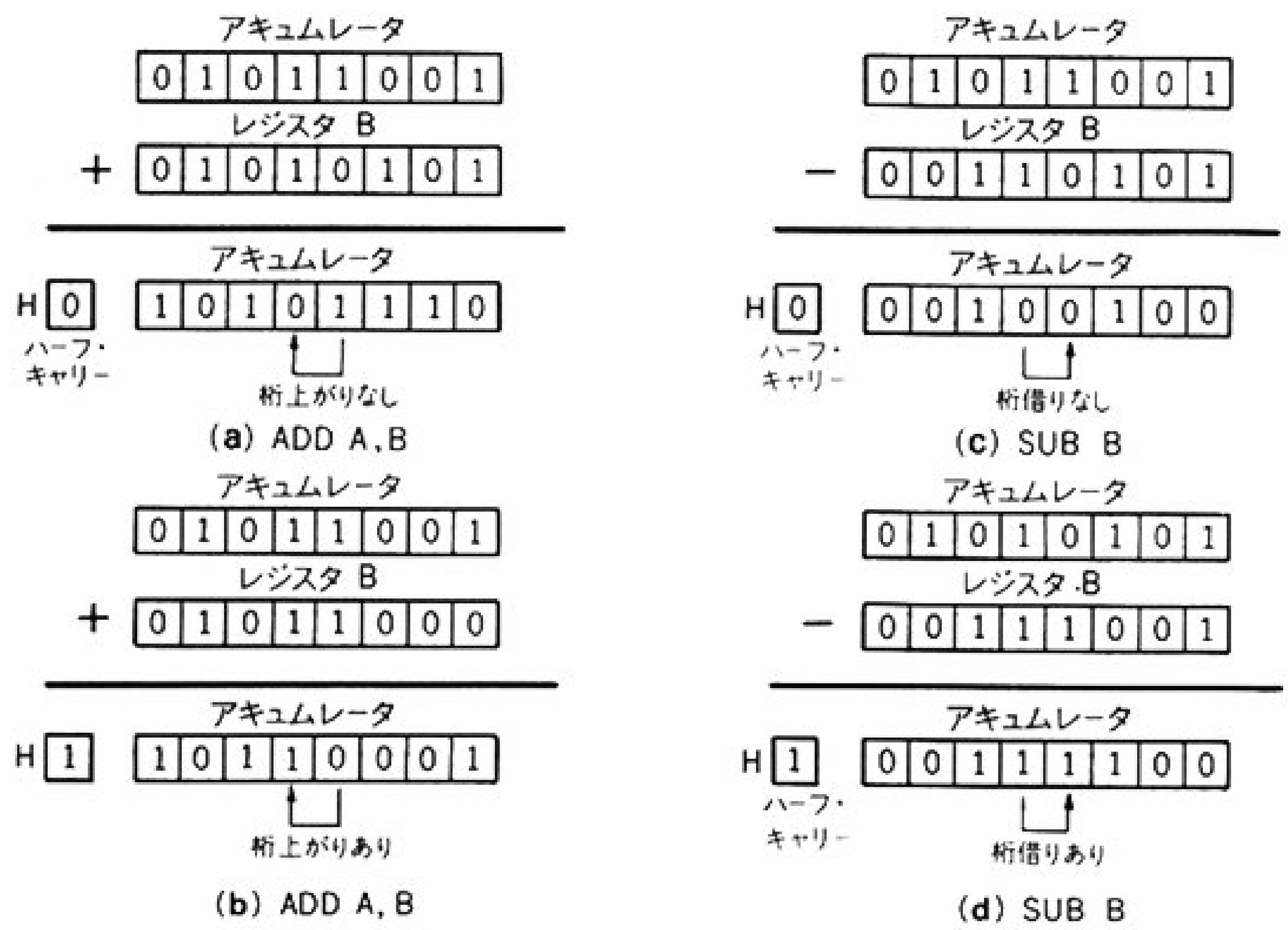
■ハーフ・キャリー・フラグ(H)

このハーフ・キャリー・フラグは加算命令(ADD，ADC)，減算命令(SUB，SBC，CP，NEG)，ならびにINC命令，およびDEC命令を実行したときに，下位4ビットから上位4ビットに対して，桁上がりまたは桁借りがあったときにセットされます．

このフラグの状態は，条件付きジャンプ命令により判定することはできませんが，10進演算における10進補正命令(DAA)が，このフラグとキャリー・フラグとNフラグの状態にしたがって，たくみに10進補正を行います．なお，DAA命令によっても，このフラグの状態は変化します．

10進演算では0〜9の数しかとり得ませんが，CPU

〈図22〉 ADD命令とSUB命令におけるハーフ・キャリーの動き



の演算は2進法で行われるため，結果は0_{16}〜F_{16}，またはキャリー/ハーフ・キャリーがセットされ，0〜3（10H〜13H）の値を示すことがあります．このような場合に，結果を10進数に戻すのが，10進補正操作です．

10進補正の基本は次のとおりです．加算命令の場合には，結果がA_{16}〜F_{16}になった場合には6を加えて桁上がりを生じさせ，キャリーまたはハーフ・キャリーがセットされ0〜3となった場合には，これらのキャリーは16進により（10ではなく）生じたものなので，やはり6を加え，かつ桁上がりを生じさせます．

減算命令では演算結果が7_{16}〜F_{16}となり，かつハーフ・キャリーまたはキャリーがセットされていて，上位桁からの借りを示しているときには，10ではなく16を借りているので，結果から6を差し引きます．

この10進補正操作を行うDAA命令の機能の詳細については，個々の命令の説明の項をごらんください．図22にADD命令，およびSUB命令におけるハーフ・キャリーの動きを示します．

■サブトラクト・フラグ(N)

このフラグは直前に実行された命令が，加算命令(ADD, ADC, INC)であるのか減算命令(SUB, SBC, CP, NEG, DEC)であるのかを示します．加算命令が実行されたときには，このフラグは“0”になり，減算命令が実行されると“1”になります．

このフラグもハーフ・キャリー・フラグと同様に，条件付きジャンプ命令により判定することはできませんが，10進補正命令(DAA)は，このフラグの状態にしたがって，10進補正操作を行います．前述のように，10進補正操作は，加算時と減算時とでは異なるからです．

＊

Z80のフラグの意味の説明のまとめとして，各命令におけるフラグの変化状態を，図23に示します．

〈図 23〉 Z 80 の命令によるフラグの変化状態

命　　令	C	Z	P/V	S	N	H	コ　メ　ン　ト
ADD A,S；ADC A,S	↕	↕	V	↕	0	↕	8ビット加算命令
SUB S；SBC A,S CP ； NEG	↕	↕	V	↕	1	↕	8ビット減算命令
AND S	0	↕	P	↕	0	1	アンド命令
OR S；XOR S	0	↕	P	↕	0	0	オア命令および排他論理和命令
INC S	●	↕	V	↕	0	↕	インクリメント命令
DEC m	●	↕	V	↕	1	↕	デクリメント命令
ADD DD,SS	↕	●	●	●	0	↕	16ビット加算命令．Hフラグはビット11から桁上がりがあったときにセットされる
ADC HL,SS	↕	↕	V	↕	0	↕	16ビット加算命令．Hフラグはビット11から桁上がりが生じたときにセットされる
SBC HL,SS	↕	↕	V	↕	1	↕	16ビット減算命令．Hフラグはビット12からの桁借りが生じたときにセットされる
RLA；RLCA；RRA；RRCA	↕	●	●	●	0	0	8080コンパチブルなローテイト命令
RL m；RLC m RR m；RRC m SLA m；SRA m；SRL m	↕	↕	P	↕	0	0	Z80固有のローテイト/シフト命令
RLD；RRD	●	↕	P	↕	0	0	4ビット単位のローテイト命令
DAA	↕	↕	P	↕	●	↕	10進補正命令
CPL	●	●	●	●	1	1	コンプリメント・アキュムレータ命令
SCF	1	●	●	●	0	0	セット・キャリー・フラグ命令
CCF	↕	●	●	●	0	↕	コンプリメント・キャリー・フラグ命令
IN r,(C)	●	↕	P	↕	0	0	Z80固有のインプット命令
INI；IND；OUTI；OUTD	●	↕	X	X	1	X	ブロック入出力命令．レジスタB=0のときにZフラグは"1"になる
INIR；INDR；OTIR；OTDR	●	1	X	X	1	X	ブロック入出力命令
LDI；LDD	●	●	↕	●	0	0	メモリ間のブロック移動命令．レジスタBC=0のときにP/V=0となる
LDIR；LDDR	●	●	0	●	0	0	メモリ間のブロック移動命令
CPI；CPIR；CPD；CPDR	●	↕	↕	↕	1	↕	ブロック・サーチ命令．A=(HL)のときZ="1"．レジスタBC=0のときP/V=0
LD　A,I；LD A,R	●	↕	IFF2	↕	0	0	EI状態のとき，P/V=1．DI状態のとき，P/V=0
BIT b,S	●	↕	X	X	0	1	テストすべきビット(b)を反転したものがZフラグにセットされる

↕：演算結果により，フラグの状態が影響を受ける
●：フラグは変化しない
X：フラグの状態は不定である(Don't Care)
V：P/Vフラグはオーバフローの意味を持つ
P：P/Vフラグはパリティの意味を持つ

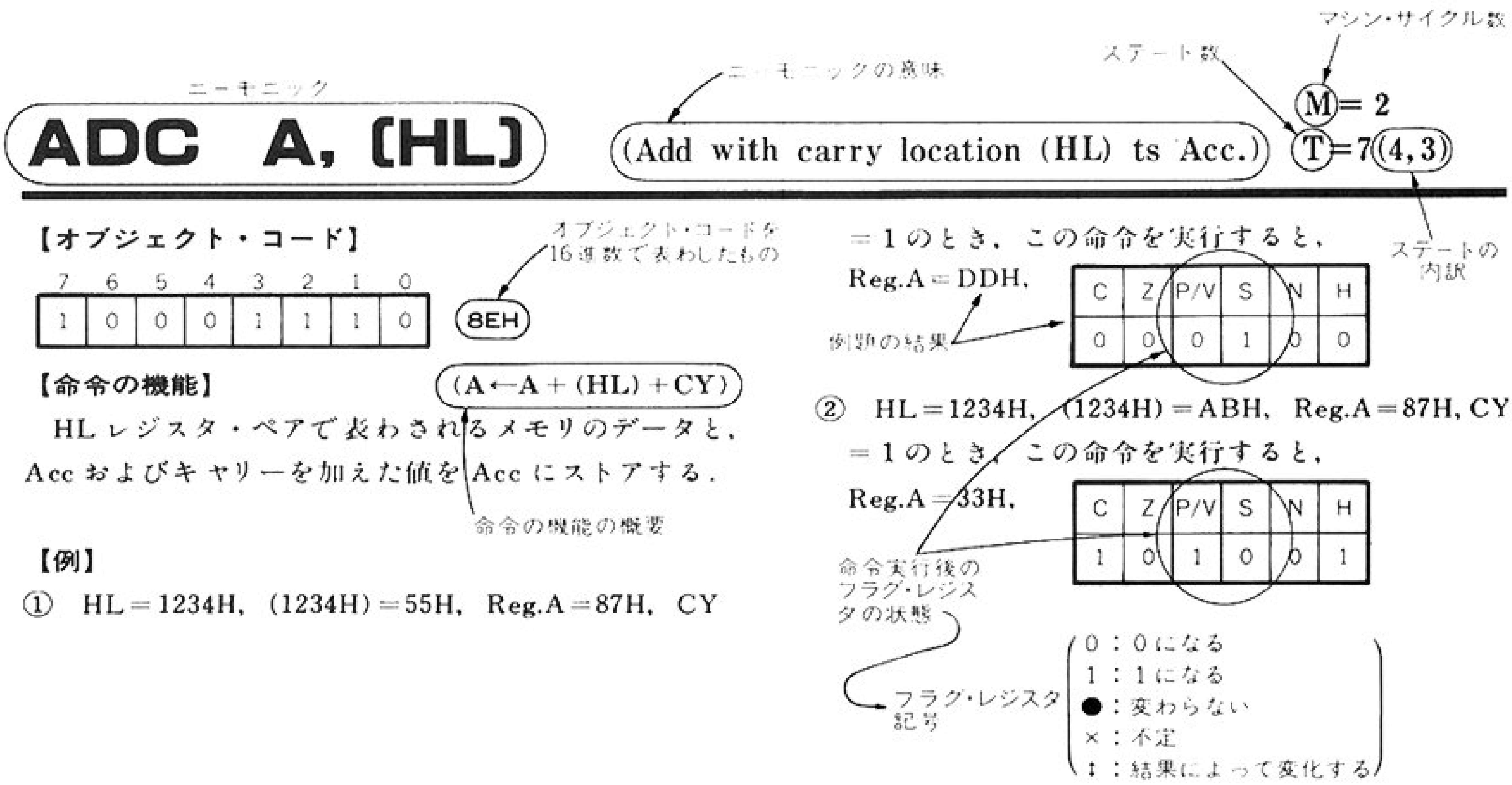
Z 80 命令の機能

72ページから，Z 80 CPUのインストラクションのすべてについて，アルファベット順に整理して詳解してあります．それぞれ，図1の例のようにビット・イメージによるオブジェクト・コードと，それを16進数で表わしたもの，命令の動作機能についての解説，およびできる限り使用例を示すようにし，フラグの変化があるものについてはそれも図示するようにしました．

オブジェクト・コードの16進数による表記は，ハンド・アセンブル時にも役立つことと思いますが，命令としての値やディスプレースメントなどの任意値となるものについては記してありません．

なお，命令の表記，あるいは機能説明中で用いた記号，ならびに略号の各意味は表1に示すとおりです．また，命令群別による索引を次頁に示します．

〈図1〉命令解説の見かた



〈表1〉使用した記号，略号の意味

A	アキュムレータ
B	レジスタB
C	レジスタC
D	レジスタD
E	レジスタE
H	レジスタH
L	レジスタL
F	フラグ・レジスタ
Reg.	レジスタ(ex.Reg.B = レジスタB)
BC	レジスタ・ペアBC
DE	レジスタ・ペアDE
HL	レジスタ・ペアHL
LX	インデックス・レジスタIX
IY	インデックス・レジスタIY
PC	プログラム・カウンタ

SP	スタック・ポインタ
R	リフレッシュ・レジスタ
I	インタラプト・レジスタ
ss	レジスタ・ペア(BC，DE，HL，SP)
r	レジスタ(B，C，D，E，H，L)
b	ビット番号
∧	論理積(AND)
∨	論理和(OR)
⊕	排他論理和(XOR)
n	16進数(8ビット)
nn	16進数(16ビット)
CY	キャリー・フラグ
d	ディスプレースメント
( )	カッコ内の値が示すアドレスのデータ

命 令 群 別 索 引

ADC A, (HL) (Add with carry location (HL) to Acc.) M=2 T=7(4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	0	0	0	1	1	1	0	8EH

【命令の機能】 (A←A+(HL)+CY)

HLレジスタ・ペアで表わされるメモリのデータと、Accおよびキャリーを加えた値をAccにストアする。

【例】

① HL=1234H, (1234H)=55H, Reg.A=87H, CY=1のとき、この命令を実行すると、

Reg.A=DDH,

C	Z	P/V	S	N	H
0	0	0	1	0	0

② HL=1234H, (1234H)=ABH, Reg.A=87H, CY=1のとき、この命令を実行すると、

Reg.A=33H,

C	Z	P/V	S	N	H
1	0	1	0	0	1

ADC A, (IX+d) (Add with carry location (IX+d) to Acc.) M=5 T=19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	0	0	0	1	1	1	0	8EH
←			d				→	XXH

【命令の機能】 (A←A+(IX+d)+CY)

インデックス・レジスタIXの内容と、ディスプレースメントdの和が示すアドレスのメモリ・データとAccおよびキャリーを加えた値をAccにストアする。

【例】

① IX=2345H, d=67H, (23ACH)=89H, Reg.A=12H, CY=1のときに、この命令を実行すると、

Reg.A=9CH,

C	Z	P/V	S	N	H
0	0	0	1	0	0

(注; 2345H+0067H=23ACH)

② IX=2345H, d=EFH, (2334H)=89H, Reg.A=ABH, CY=1のときに、この命令を実行すると、

Reg.A=35H,

C	Z	P/V	S	N	H
1	0	1	0	0	1

(注; 2345H+FFEFH=2334H)

ADC A, (IY+d) (Add with carry location (IY+d) to Acc.) M=5 T=19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	0	0	0	1	1	1	0	8EH
←			d				→	XXH

【命令の機能】 (A←A+(IY+d)+CY)

インデックス・レジスタIYの内容と、ディスプレースメントdの和が示すアドレスのメモリ・データと、Accおよびキャリーを加えた値をAccにストアする。

【例】

① IY=2345H, d=67H, (23ACH)=89H, Reg.A=12H, CY=1のときに、この命令を実行すると、

Reg.A=9CH,

C	Z	P/V	S	N	H
0	0	0	1	0	0

(注; 2345H+67H=23ACH)

② IY=2345H, d=EFH, (2334H)=89H, Reg.A=ABH, CY=1のときに、この命令を実行すると、

Reg.A=35H,

C	Z	P/V	S	N	H
1	0	1	0	0	1

(注; 2345H+FFEFH=2334H)

ADC　A, r

(Add with carry Register r to Acc.)　M＝1　T＝4

【オブジェクト・コード】

7	6	5	4	3	2	1	0
1	0	0	0	1	←r→		

r	r の値	ニーモニック	オブジェクト・コード
B	000	ADC A,B	88H
C	001	ADC A,C	89H
D	010	ADC A,D	8AH
E	011	ADC A,E	8BH
H	100	ADC A,H	8CH
L	101	ADC A,L	8DH
A	111	ADC A,A	8FH

【命令の機能】　(A←A＋r＋CY)

レジスタr(A～L)の内容と，Accおよびキャリーを加えた値をAccにストアする．

【例】

① Reg.A＝EEH, Reg.B＝11H, CY＝1のときに，この命令(ADC　A, B)を実行すると，

Reg.A＝00H,

C	Z	P/V	S	N	H
1	1	0	0	0	1

② Reg.A＝EEH, CY＝1のときに，この命令(ADC A, A)を実行すると，

Reg.A＝DDH,

C	Z	P/V	S	N	H
1	0	0	1	0	1

ADC　A, n

(Add with carry value n to Acc.)　M＝2　T＝7(4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	1	1	0	CEH
←n→								

【命令の機能】　(A←A＋n＋CY)

イミーディエイト・バリューnとキャリーをAccに加える．

【例】

① Reg.A＝EEH, CY＝1のとき，ADC　A, 11H命令を実行すると，

Reg.A＝00H,

C	Z	P/V	S	N	H
1	1	0	0	0	1

② Reg.A＝55H, CY＝1のとき，ADC　A, 66H命令を実行すると，

Reg.A＝BCH,

C	Z	P/V	S	N	H
0	0	1	1	0	0

ADC　HL, ss

(Add with carry register pair to HL)　M＝4　T＝15(4,4,4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	s	s	1	0	1	0	

ss	ssの値	ニーモニック	オブジェクト・コード
BC	00	ADC HL, BC	4AH
DE	01	ADC HL, DE	5AH
HL	10	ADC HL, HL	6AH
SP	11	ADC HL, SP	7AH

【命令の機能】　(HL←HL＋ss＋CY)

ssで指定されるレジスタ・ペアの内容と，HLおよびキャリーを加え，結果をHLにストアする．

【例】

① HL＝3A34H, BC＝5A78H, CY＝1のときにADC　HL, BC命令を実行すると，

HL＝94ADH,

C	Z	P/V	S	N	H
0	0	1	1	0	1

② HL＝1234H, CY＝1のときにADC　HL, HL命令を実行すると，

HL＝2469H,

C	Z	P/V	S	N	H
0	0	0	0	0	0

ADD A, (HL)

(Add location (HL) to Acc.)

M＝2
T＝7(4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	0	0	0	0	1	1	0	86H

【命令の機能】　(A←A＋(HL))

HLレジスタ・ペアで表わされるメモリのデータとAccの内容を加え，結果をAccにストアする．

【例】

① HL＝1234H，(1234H)＝55H，Reg. A＝27Hのとき，この命令を実行すると，

Reg.A＝7CH，

C	Z	P/V	S	N	H
0	0	0	0	0	0

② HL＝1234H，(1234H)＝ABH，Reg.A＝87Hのとき，この命令を実行すると，

Reg.A＝32H，

C	Z	P/V	S	N	H
1	0	1	0	0	1

ADD A, (IX+d)

(Add location (IX+d) to Acc.)

M＝5
T＝19(4,4,3,5,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	0	0	0	0	1	1	0	86H
←──			d				──→	

【命令の機能】　(A←A＋(IX＋d))

インデックス・レジスタIXと，ディスプレースメントdの和が示すアドレスのメモリ・データをAccに加え，結果をAccにストアする．

【例】

① IX＝2345H，d＝28H，(236DH)＝89H，Reg.A＝12Hのとき，この命令を実行すると，

Reg.A＝9BH，

C	Z	P/V	S	N	H
0	0	0	1	0	0

(注；2345H＋0028H＝236DH)

② IX＝2345H，d＝80H，(22C5H)＝89H，Reg.A＝ABHのとき，この命令を実行すると，

Reg.A＝34H，

C	Z	P/V	S	N	H
1	0	1	0	0	1

(注；2345H＋FF80H＝22C5H)

ADD A, (IY+d)

(Add location (IY+d) to Acc.)

M＝5
T＝19(4,4,3,5,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	0	0	0	0	1	1	0	86H
←──			d				──→	

【命令の機能】　(A←A＋(IY＋d))

インデックス・レジスタIYと，ディスプレースメントdの和が示すアドレスのメモリ・データとAccを加え，結果をAccにストアする．

【例】

① IY＝ABCDH，d＝67H，(AC34H)＝89H，Reg.A＝12Hのとき，この命令を実行すると，

Reg. A＝9BH，

C	Z	P/V	S	N	H
0	0	0	1	0	0

(注；ABCDH＋0067H＝AC34H)

② IY＝ABCDH，d＝8FH，(AB5CH)＝89H，Reg.A＝ABHのとき，この命令を実行すると，

Reg.A＝34H，

C	Z	P/V	S	N	H
1	0	1	0	0	1

(注；ABCDH＋FF8FH＝AB5CH)

ADD A, r

(Add register r to Acc.)

M＝1
T＝4

【オブジェクト・コード】

7	6	5	4	3	2	1	0
1	0	0	0	0	←	r	→

r	r の値	ニーモニック	オブジェクト・コード
B	0 0 0	ADD A, B	8 0 H
C	0 0 1	ADD A, C	8 1 H
D	0 1 0	ADD A, D	8 2 H
E	0 1 1	ADD A, E	8 3 H
H	1 0 0	ADD A, H	8 4 H
L	1 0 1	ADD A, L	8 5 H
A	1 1 1	ADD A, A	8 7 H

【命令の機能】 (A←A＋r)

レジスタ r (A～L)の内容と Acc を加え、結果を Acc にストアする。

【例】

① Reg.A＝EEH, Reg.B＝11H のときに、この命令(ADD A, B)を実行すると、
Reg.A＝FFH,

C	Z	P/V	S	N	H
0	0	0	1	0	0

② Reg.A＝EEH のときに、この命令(ADD A, A)を実行すると、
Reg.A＝DCH,

C	Z	P/V	S	N	H
1	0	0	1	0	1

ADD A, n

(Add value n to Acc.)

M＝2
T＝7(4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	0	1	1	0	C6H
←			n				→	

【命令の機能】 (A←A＋n)

イミーディエイト・バリュー n を Acc に加える。

【例】

① Reg.A＝3AH のときに ADD A, 38H 命令を実行すると、
Reg.A＝72H,

C	Z	P/V	S	N	H
0	0	0	0	0	1

② Reg.A＝55H のときに、ADD A, F6H 命令を実行すると、
Reg.A＝4BH,

C	Z	P/V	S	N	H
1	0	0	0	0	0

ADD HL, ss

(Add register pair to HL)

M＝3
T＝11(4,4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0
0	0	s	s	1	0	0	1

s s	ssの値	ニーモニック	オブジェクト・コード
B C	0 0	ADD HL, BC	0 9 H
D E	0 1	ADD HL, DE	1 9 H
H L	1 0	ADD HL, HL	2 9 H
S P	1 1	ADD HL, SP	3 9 H

【命令の機能】 (HL←HL＋ss)

ss で指定されるレジスタ・ペアの内容と HL を加え、結果を HL にストアする。

【例】

① HL＝3A34H, BC＝5A78H のときに、ADD HL, BC 命令を実行すると、
HL＝94ACH,

C	Z	P/V	S	N	H
0	●	●	●	0	1

② HL＝8765H のとき、ADD HL, HL 命令を実行すると、
HL＝0ECAH,

C	Z	P/V	S	N	H
1	●	●	●	0	0

ADD IX, ss

(Add register pair to IX)

M＝4
T＝15(4,4,4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
0	0	s	s	1	0	0	1	

ss	ssの値	ニーモニック	オブジェクト・コード
BC	00	ADD IX, BC	09H
DE	01	ADD IX, DE	19H
IX	10	ADD IX, IX	29H
SP	11	ADD IX, SP	39H

【命令の機能】　(IX←IX＋ss)

レジスタ・ペアの内容をインデックス・レジスタIXに加える．

【例】

① IX＝1234H，BC＝5678Hのとき，ADD IX，BC命令を実行すると，
IX＝68ACH，

C	Z	P/V	S	N	H
0	●	●	●	0	0

② IX＝8765H，DE＝FEDCHのとき，ADD IX，DE命令を実行すると，
IX＝8641H，

C	Z	P/V	S	N	H
1	●	●	●	0	1

ADD IY, ss

(Add register pair to IY)

M＝4
T＝15(4,4,4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
0	0	s	s	1	0	0	1	

ss	ssの値	ニーモニック	オブジェクト・コード
BC	00	ADD IY, BC	09H
DE	01	ADD IY, DE	19H
IY	10	ADD IY, IY	29H
SP	11	ADD IY, SP	39H

【命令の機能】　(IY←IY＋ss)

レジスタ・ペアの内容をインデックス・レジスタIYに加える．

【例】

① IY＝1234H，BC＝5678Hのとき，ADD IY，BC命令を実行すると，
IY＝68ACH，

C	Z	P/V	S	N	H
0	●	●	●	0	0

② IY＝FEDCHのとき，ADD IY，IY命令を実行すると，
IY＝FDB8H，

C	Z	P/V	S	N	H
1	●	●	●	0	1

AND (HL)

(Lgical"AND" of location (HL) and Acc.)

M＝2
T＝7(4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	0	1	0	0	1	1	0	A6H

【命令の機能】　(A←A∧(HL))

HLレジスタ・ペアで表わされるメモリの内容と，Accとの論理積をとり，結果をAccにストアする．

【例】

① HL＝4567H，(4567H)＝83H(1000 0011)，Reg.A＝CFH(1100 1111)であるときに，この命令を実行すると，
Reg.A＝83H，

C	Z	P/V	S	N	H
0	0	0	1	0	1

② HL＝4567H，(4567H)＝55H(0101 0101)，Reg.A＝AAH(1010 1010)であるときに，この命令を実行すると，
Reg.A＝00H，

C	Z	P/V	S	N	H
0	1	1	0	0	1

AND (IX+d)

(Logical "AND" of location (IX+d) and Acc.) M=5 T=19(4,4,3,5,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	0	1	0	0	1	1	0	A6H
←— d —→								

【命令の機能】 (A←A∧(IX+d))

インデックス・レジスタIXとディスプレースメントdとの和が示すメモリの内容と，Accとの論理積をとり，結果をAccにストアする．

【例】

IX=4567H，d=43H，(45AAH)=83H(1000 0011)，Reg.A=FCH(1111 1100)のときに，この命令を実行すると，

Reg.A=80H，

C	Z	P/V	S	N	H
1	0	0	0	0	1

(注; 4567H+43H=45AAH)

AND (IY+d)

(Logical "AND" of location (IY+d) and Acc.) M=5 T=19(4,4,3,5,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	0	1	0	0	1	1	0	A6H
←— d —→								

【命令の機能】 (A←A∧(IY+d))

インデックス・レジスタIYとディスプレースメントdの和で示されるメモリの内容と，Accとの論理積をとり，結果をAccにストアする．

【例】

IY=4567H，d=D6H，(453DH)=77H，Reg.A=FFHのときに，この命令を実行すると，

Reg.A=77H，

C	Z	P/V	S	N	H
0	0	1	0	0	1

(注; 4567H+FFD6H=453DH)

AND r

(Logical "AND" of register r and Acc.) M=1 T=4

【オブジェクト・コード】

7	6	5	4	3	2	1	0
1	0	1	0	0	←— r —→		

r	rの値	ニーモニック	オブジェクト・コード
B	000	AND B	A0H
C	001	AND C	A1H
D	010	AND D	A2H
E	011	AND E	A3H
H	100	AND H	A4H
L	101	AND L	A5H
A	111	AND A	A7H

【命令の機能】 (A←A∧r)

レジスタr(A～L)の内容とAccとの論理積をとり，結果をAccにストアする．

【例】

① Reg.A=77H(0111 0111)，Reg.B=CCH(1100 1100)のときに，AND B命令を実行すると，

Reg.A=44H，

C	Z	P/V	S	N	H
0	0	1	0	0	1

② Reg.A=00Hであるときに，AND A命令を実行すると，

Reg.A=00H，

C	Z	P/V	S	N	H
0	1	1	0	0	1

AND n

M＝2
(Logical "AND" of value n and Acc.) T＝7(4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	0	1	1	0	E6H
←			n				→	

【命令の機能】 (A←A∧n)

Accの内容とイミーディエイト・バリューnとの論理積をとり，結果をAccにストアする．

【例】

Acc＝CFH(1100 1111)であるときに，AND 83H命令を実行すると，

Reg.A＝83H，

C	Z	P/V	S	N	H
0	0	0	1	0	1

BIT b, (HL)

M＝3
(Test bit of location (HL)) T＝12(4,4,4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	1	←	b	→	1	1	0	

ビット	bの値	ニーモニック	オブジェクト・コード
0	000	BIT 0, (HL)	46H
1	001	BIT 1, (HL)	4EH
2	010	BIT 2, (HL)	56H
3	011	BIT 3, (HL)	5EH
4	100	BIT 4, (HL)	66H
5	101	BIT 5, (HL)	6EH
6	110	BIT 6, (HL)	76H
7	111	BIT 7, (HL)	7EH

【命令の機能】 $(Z \leftarrow \overline{(HL)_b})$

HLレジスタ・ペアが示すメモリのデータに，テストするビット(b)が立っているか否かをチェックする．

もし立っていれば(1)ゼロ・フラグ(Z)は"0"となり，立っていなければゼロ・フラグ(Z)は"1"となる．

したがってこの命令は，HLレジスタが示すメモリのデータ中のテストすべきビットを反転したものを，ゼロ・フラグにセットするのである．

【例】

① HL＝23C5H，(23C5H)＝55H(0101 0101)のときに，BIT 0，(HL)命令を実行すると，ゼロ・フラグ(Z)は"0"になる．

② HL＝23C5H，(23C5H)＝AAH(1010 1010)のときに，BIT 2，(HL)命令を実行すると，ゼロ・フラグ(Z)は"1"になる．

BIT b, (IX+d)

(Test bit of location (IX+d))　M＝5　T＝20(4,4,3,5,4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	0	0	1	0	1	1	CBH
←d→								
0	1	←b→			1	1	0	

ビット	b の値	ニーモニック	オブジェクト・コード
0	0 0 0	BIT 0, (IX+d)	4 6 H
1	0 0 1	BIT 1, (IX+d)	4 E H
2	0 1 0	BIT 2, (IX+d)	5 6 H
3	0 1 1	BIT 3, (IX+d)	5 E H
4	1 0 0	BIT 4, (IX+d)	6 6 H
5	1 0 1	BIT 5, (IX+d)	6 E H
6	1 1 0	BIT 6, (IX+d)	7 6 H
7	1 1 1	BIT 7, (IX+d)	7 E H

【命令の機能】　$(Z \leftarrow \overline{(IX+d)}_b)$

インデックス・レジスタIXと，ディスプレースメントdの和によって示されるメモリのデータのビットが立っているか否かをチェックする．

もしビットが立っている("1")ときには，ゼロ・フラグ(Z)は"0"になり，ビットが立っていない("0")ときには，ゼロ・フラグ(Z)は"1"になる．

【例】

① IX＝2345H, d＝55H, (239AH)＝55H (0101 0101)のときに，BIT 2, (IX+d)命令を実行すると，ゼロ・フラグ(Z)は"0"になる．

(注; 2345H＋55H＝239AH)

② IX＝23C5H, d＝C8H, (230DH)＝AAH(1010 1010)のときに，BIT 6, (IX+d)命令を実行すると，ゼロ・フラグ(Z)は"1"になる．

(注; 23C5H＋FFC8H＝230DH)

BIT b, (IY+d)

(Test bit of location (IY+d))　M＝5　T＝20(4,4,3,5,4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	0	0	1	0	1	1	CBH
←d→								
0	1	←b→			1	1	0	

ビット	b の値	ニーモニック	オブジェクト・コード
0	0 0 0	BIT 0, (IY+d)	4 6 H
1	0 0 1	BIT 1, (IY+d)	4 E H
2	0 1 0	BIT 2, (IY+d)	5 6 H
3	0 1 1	BIT 3, (IY+d)	5 E H
4	1 0 0	BIT 4, (IY+d)	6 6 H
5	1 0 1	BIT 5, (IY+d)	6 E H
6	1 1 0	BIT 6, (IY+d)	7 6 H
7	1 1 1	BIT 7, (IY+d)	7 E H

【命令の機能】　$(Z \leftarrow \overline{(IY+d)}_b)$

インデックス・レジスタIYとディスプレースメントdの和により示される，メモリのデータのビットが立っているか否かをチェックする．

もしビットが立っている("1")ときは，ゼロ・フラグ(Z)は"0"になり，ビットが立っていない("0")ときは，ゼロ・フラグ(Z)は"1になる．

【例】

① IY＝2345H, d＝55H, (239AH)＝55H (0101 0101)のときに，BIT 2, (IY+d)命令を実行すると，ゼロ・フラグ(Z)は"0"になる．

(注; 2345H＋55H＝239AH)

② IY＝23C5H, d＝C8H, (230DH)＝AAH(1010 1010)のときに，BIT 6, (IY+d)命令を実行すると，ゼロ・フラグ(Z)は"1"になる．

(注; 23C5H＋FFC8H＝230DH)

BIT b, r

(Test bit of register r)　M＝2　T＝8(4,4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	1	←b→			←r→			

		ビットb							
		0	1	2	3	4	5	6	7
レジスタr	B	40	48	50	58	60	68	70	78
	C	41	49	51	59	61	69	71	79
	D	42	4A	52	5A	62	6A	72	7A
	E	43	4B	53	5B	63	6B	73	7B
	H	44	4C	54	5C	64	6C	74	7C
	L	45	4D	55	5D	65	6D	75	7D
	A	47	4F	57	5F	67	6F	77	7F

【命令の機能】　$(Z \leftarrow \overline{r_b})$

レジスタrのビットbをチェックし，ビットbが立っていれば("1")ゼロ・フラグ(Z)を"0"にし，ビットbが立っていなければ("0")，ゼロ・フラグを"1"にする．

つまり，レジスタrのビットbを反転したものを，ゼロ・フラグ(Z)にセットする．

【例】

① Reg.H＝78H(0111 1000)であるとき，BIT 4，H命令を実行すると，ゼロ・フラグ(Z)は"0"になる．

② Reg.A＝AAH(1010 1010)であるときに，BIT 0，A命令を実行すると，ゼロ・フラグ(Z)は"1"になる．

CALL cc, nn

条件成立時 (M＝5, T＝17(4,3,4,3,3))，条件不成立時 (M＝3, T＝10(4,3,3))

(Call subroutine at location nn if condition cc is true)

【オブジェクト・コード】

7	6	5	4	3	2	1	0
1	1	←cc→			1	0	0
←n→							
←n→							

条件	cc	ccの値	ニーモニック	OPコード
Non Zero (Z＝0)	NZ	000	CALL NZ, nn	C4H
Zero (Z＝1)	Z	001	CALL Z, nn	CCH
Non Carry (C＝0)	NC	010	CALL NC, nn	D4H
Carry (C＝1)	C	011	CALL C, nn	DCH
Parity Odd (P/V＝0)	PO	100	CALL PO, nn	E4H
Parity Even (P/V＝1)	PE	101	CALL PE, nn	ECH
Positive (S＝0)	P	110	CALL P, nn	F4H
Negative (S＝1)	M	111	CALL M, nn	FCH

【命令の機能】(条件が成立すれば，$(SP-1) \leftarrow PC_H$，$(SP-2) \leftarrow PC_L$，$PC \leftarrow nn$，$SP \leftarrow SP-2$)

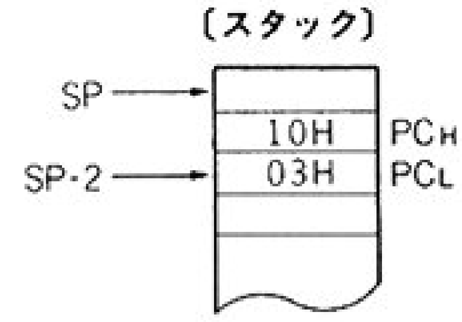
判定条件(cc)が成立しないときには，次の命令を実行する．判定条件が成立したときには，PC_Hを(SP−1)にストアし，PC_Lを(SP−2)にストアし，アドレスnnへジャンプする．

スタック・ポインタ(SP)は2だけ減じられる．フラグは変化しない．

【例】

① ゼロ・フラグ(Z)が立っている("1")ときに，1000H番地の命令，CALL Z，1234Hを実行すると，(SP−1)に10Hをストアし，(SP−2)に03Hをストアした後に，1234H番地にジャンプする．スタック・ポインタ(SP)は2だけ減じられる．

〔アドレス〕	〔命令〕
1000H	CCH
1001H	34H
1002H	12H
1003H	次の命令
⋮	
1234H	←この命令を実行する



② キャリー・フラグ(C)が立っている("1")ときに，1000H番地の命令，CALL NC，1234Hを実行すると，条件が成立しないのでジャンプは行わず，次の命令(1003H番地)を実行する．スタック・ポインタ(SP)は変化しない．

〔アドレス〕	〔命令〕
1000H	D4H
1001H	34H
1002H	12H
1003H	次の命令 ←この命令を実行する

CALL nn (Unconditional Call to Subroutine at nn)

M＝5
T＝17(4,3,4,3,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	1	0	1	CDH
←			n				→	
←			n				→	

【命令の機能】 $\begin{pmatrix} (SP-1) \leftarrow PC_H,\ (SP-2) \leftarrow PC_L, \\ PC \leftarrow nn,\ SP \leftarrow SP-2 \end{pmatrix}$

PC_H を(SP−1)，PC_L を(SP−2)にストアした後に無条件に，アドレス nn へジャンプする．

スタック・ポインタ(SP)は2だけ減じられる．フラグは変化しない．

【例】

1000H 番地の命令，CALL 1234H 命令を実行すると，(SP−1)に 10H をストアし，(SP−2)に 03H をストアした後に，1234H 番地にジャンプする．

スタック・ポインタ(SP)は2だけ減じられる．

CCF (Complement Carry Flag)

M＝1
T＝4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	1	1	1	1	1	1	3FH

【命令の機能】 $(C \leftarrow \overline{C})$

フラグ・レジスタ中のキャリー・フラグが反転される．H フラグには，命令実行前のキャリー・フラグの状態がセーブされる．

【例】

① キャリー・フラグ＝0のときに，この命令を実行すると，

C	Z	P/V	S	N	H
1	●	●	●	0	0

② キャリー・フラグ＝1のときに，この命令を実行すると，

C	Z	P/V	S	N	H
0	●	●	●	0	1

CP (HL) (Compare Location(HL) with Acc.)

M＝2
T＝7(4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	0	1	1	1	1	1	0	BEH

【命令の機能】 (A－(HL))

Acc の内容から HL レジスタ・ペアが示すアドレスのデータを減じ，結果をフラグ・レジスタにセットする．Acc の内容は変化しない．

【例】

① HL＝2345H，(2345H)＝55H，Reg.A＝56H であるときに，この命令を実行すると，

C	Z	P/V	S	N	H
0	0	0	0	1	0

② HL＝2345H，(2345H)＝56H，Reg.A＝55H であるときに，この命令を実行すると，

C	Z	P/V	S	N	H
1	0	0	1	1	1

③ HL＝2345H，(2345H)＝55H，Reg.A＝55H であるときに，この命令を実行すると，

C	Z	P/V	S	N	H
0	1	0	0	1	0

④ HL＝2345H，(2345H)＝7FH，Reg.A＝88H のときに，この命令を実行すると，

C	Z	P/V	S	N	H
0	0	1	0	1	1

⑤ HL＝2345H，(2345H)＝80H，Reg.A＝88H のときに，この命令を実行すると，

C	Z	P/V	S	N	H
0	0	0	0	1	0

CP (IX+d)

(Compare Location (IX+d) with Acc.)

M=5
T=19(4,4,3,5,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	0	1	1	1	1	1	0	BEH
←—			d				—→	

【命令の機能】　(A－(IX＋d))

Accの内容から，インデックス・レジスタIXの内容とディスプレースメントdを加えた値が示すアドレスの内容を減じ，結果をフラグ・レジスタにセットする．Accの内容は変化しない．

【例】

① Reg.A＝78H，IX＝2000H，d＝40H，(2040H)＝57Hであるときに，この命令を実行すると，

C	Z	P/V	S	N	H
0	0	0	0	1	0

② Reg.A＝98H，IX＝2100H，d＝80H，(2080H)＝C6Hであるときに，この命令を実行すると，

C	Z	P/V	S	N	H
1	0	0	1	1	0

(注; 2100H＋FF80H＝2080H)

CP (IY+d)

(Compare Location (IY+d) with Acc.)

M=5
T=19(4,4,3,5,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	0	1	1	1	1	1	0	BEH
←—			d				—→	

【命令の機能】　(A－(IY＋d))

Accの内容から，インデックス・レジスタIYとディスプレースメントdを加えた値が示すアドレスの内容を減じ，結果をフラグ・レジスタにセットする．Accの内容は変化しない．

【例】

① Reg.A＝78H，IY＝2000H，d＝40H，(2040H)＝57Hであるときに，この命令を実行すると，

C	Z	P/V	S	N	H
0	0	0	0	1	0

② Reg.A＝98H，IY＝2100H，d＝80H，(2080H)＝C6Hであるときに，この命令を実行すると，

C	Z	P/V	S	N	H
1	0	0	1	1	0

(注; 2100H＋FF80H＝2080H)

CP r

(Compare Register r with Acc.)

M = 1
T = 4

【オブジェクト・コード】

7	6	5	4	3	2	1	0
1	0	1	1	1	←	r	→

r	r の値	ニーモニック	オブジェクト・コード
B	0 0 0	CP B	B 8
C	0 0 1	CP C	B 9
D	0 1 0	CP D	B A
E	0 1 1	CP E	B B
H	1 0 0	CP H	B C
L	1 0 1	CP L	B D
A	1 1 1	CP A	B F

【命令の機能】　　(A－r)

Accの内容から，レジスタrの内容を減じ，結果をフラグ・レジスタにセットする．Accの内容は変化しない．

【例】

① Reg.A＝78H，Reg.B＝47Hのとき，CP B命令を実行すると，

C	Z	P/V	S	N	H
0	0	0	0	1	0

② Reg.A＝47H，Reg.L＝78Hのとき，CP L命令を実行すると，

C	Z	P/V	S	N	H
1	0	0	1	1	1

CP n

(Compare n with Acc.)

M = 2
T = 4,3

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	1	0	FEH
←			n				→	

【命令の機能】　　(A－n)

Accの内容から，イミーディエイト・データnを減じ，結果をフラグ・レジスタにセットする．

【例】

① Reg.A＝78Hのときに，CP 47H命令を実行すると，

C	Z	P/V	S	N	H
0	0	0	0	1	0

② Reg.A＝47Hのときに，CP 78H命令を実行すると，

C	Z	P/V	S	N	H
1	0	0	1	1	1

③ Reg.A＝AAHのときに，CP AAH命令を実行すると，

C	Z	P/V	S	N	H
0	1	0	0	1	0

CPD

(Compare location (HL) with Acc. and decrement HL & BC)

M = 4
T = 16 (4, 4, 3, 5)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	0	1	0	0	1	A9H

【命令の機能】 (A − (HL), BC←BC − 1, HL←HL − 1)

Accの内容から，レジスタ・ペアHLが示すメモリの内容を減じ，結果をフラグ・レジスタにセットする．

レジスタ・ペアBCおよびHLの内容は1だけ減じられる．Accの内容は変化しない．

【例】

① Reg.A＝55H，HL＝1234H，BC＝0100H，(1234H)＝55Hのときに，この命令を実行すると，

HL＝1233H，
BC＝00FFH

C	Z	P/V	S	N	H
●	1	1	0	1	0

② Reg.A＝55H，HL＝1234H，BC＝0001H，(1234H)＝AAHのときに，この命令を実行すると，

HL＝1233H，
BC＝0000H

C	Z	P/V	S	N	H
●	0	0	1	1	1

CPDR

BC≠0で A≠(HL)時 (M＝5, T＝21 (4, 4, 3, 5, 5))　BC＝0，または A＝(HL)時 (M＝4, T＝16 (4, 4, 3, 5))

(Compare location (HL) with Acc., decrement BC & HL and repeat until BC＝0.)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	1	1	0	0	1	B9H

【命令の機能】 (A − (HL), BC←BC − 1, HL←HL − 1)

Accの内容からHLで示されるメモリの内容が減じられ，結果がフラグ・レジスタにセットされる．BCおよびHLの内容は1だけ減じられる．

この命令はBC＝0となるか，A＝(HL)となるまで繰り返される．Accの内容は変化しない．

【例】

① Reg.A＝55H，HL＝20FFH，BC＝0100Hでメモリの内容が次のとおりであるとき，

20FFH	20FEH	20FDH		2001H	2000H
00H	AAH	55H	～	XX	XX

この命令を実行すると，

HL＝20FCH，
BC＝00FDH

C	Z	P/V	S	N	H
●	1	1	0	1	0

② Reg.A＝55H，HL＝20FFH，BC＝0100Hであり，2000H番地～20FFH番地に55Hが存在しない場合，この命令を実行すると，

HL＝1FFFH，
BC＝0000H

C	Z	P/V	S	N	H
●	0	0	0	1	0

CPI

(Compare location (HL) with Acc. and increment HL, decrement BC)　M＝4　T＝16(4,4,3,5)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	0	0	0	0	1	A1H

【命令の機能】 (A－(HL), HL＝HL＋1, BC＝BC－1)

Accの内容からHLで示されるメモリの内容が減じられ，結果がフラグ・レジスタにセットされる．HLには1が加えられ，BCからは1が減じられる．Accの内容は変化しない．

【例】

① Reg.A＝55H，HL＝2345H，BC＝0040H，(HL)＝66Hのときに，この命令を実行すると，

HL＝2346H,
BC＝003FH

C	Z	P/V	S	N	H
●	0	1	1	1	1

② Reg.A＝55H，HL＝2345H，BC＝0040H，(HL)＝55Hのときに，この命令を実行すると，

HL＝2346H,
BC＝003FH

C	Z	P/V	S	N	H
●	1	1	0	1	0

CPIR

BC≠0で A≠(HL)時 (M＝5 T＝21(4,4,3,5,5))　BC＝0，または A＝(HL)時 (M＝4 T＝16(4,4,3,5))

(Compare location (HL) with Acc., increment HL, decrement BC, and repeat until BC＝0)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	1	0	0	0	1	B1H

【命令の機能】 (A－(HL), HL＝HL＋1, BC＝BC－1)

Accの内容からHLで示されるメモリの内容を減じ，結果をフラグ・レジスタにセットする．HLには1が加えられ，BCからは1が減じられる．

命令はA＝(HL)，またはBC＝0となるまで繰り返される．

【例】

① Reg.A＝FDH，HL＝2345H，BC＝0100Hで，メモリの内容が次のようになっているとき，

2345H	2346H	2347H		…	2442H	2443H	2444H
00H	02H	03H		…	FDH	FEH	FFH

この命令を実行すると，

HL＝2443H,
BC＝0002H

C	Z	P/V	S	N	H
●	1	1	0	1	0

② HL＝2345H，BC＝0100Hでメモリ上にアキュムレータと一致するデータがないとき，この命令を実行すると，

HL＝2445H,
BC＝0000H

C	Z	P/V	S	N	H
●	0	0	X	1	X

CPL

(Complement accumulator)　M＝1　T＝4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	1	0	1	1	1	1	2FH

【命令の機能】　$(A \leftarrow \overline{A})$

アキュムレータの各ビットの値を反転する．

【例】

① Reg.A＝55H(0101 0101)のとき，この命令を実行すると，

Reg.A＝AAH(1010 1010),

C	Z	P/V	S	N	H
●	●	●	●	1	1

② Reg.A＝FFH(1111 1111)のとき，この命令を実行すると，

Reg.A＝00H(0000 0000),

C	Z	P/V	S	N	H
●	●	●	●	1	1

(注; ゼロ・フラグ(Z)も変化しない)

DAA

(Decimal Adjust Accumulator)

M= 1
T= 4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	1	0	0	1	1	1	27H

【命令の機能】

この命令は，フラグ・レジスタの状態にしたがって，アキュムレータの内容に対して 10 進補正を行う．

加算命令(ADD，ADC，INC)，および減算命令(SUB，SBC，DEC，NEG)における，補正の方法は次のとおり．両者の区別は N フラグにより行われる．

演 算	DAA命令を実行する前の Cフラグ	アキュムレータの上位4ビット	DAA命令を実行する前の Hフラグ	アキュムレータの下位4ビット	加算される数	DAA命令を実行した後の Cフラグ
	0	0～9	0	0～9	00	0
	0	0～8	0	A～F	06	0
ADD	0	0～9	1	0～3	06	0
	0	A～F	0	0～9	60	1
ADC	0	9～F	0	A～F	66	1
	0	A～F	1	0～3	66	1
INC	1	0～2	0	0～9	60	1
	1	0～2	0	A～F	66	1
	1	0～3	1	0～3	66	1
SUB	0	0～9	0	0～9	00	0
SBC	0	0～8	1	6～F	FA	0
DEC	1	7～F	0	0～9	A0	1
NEG	1	6～F	1	6～F	9A	1

【例】

① 10 進数の 59 と 38 を加える場合，

Reg.A=59H のとき，ADD A，38H 命令を実行すると，

	0	1	0	1	1	0	0	1	Reg.A
+	0	0	1	1	1	0	0	0	n
	1	0	0	1	0	0	0	1	Reg.A

C	Z	P/V	S	N	H
0	0	1	1	0	1

次に DAA 命令を実行すると，H=1 であるので Reg.A には 06H が加えられ，結果は 97H となる(59+38−97)．

フラグ・レジスタの値は次のようになる．

C	Z	P/V	S	N	H
0	0	0	1	0	0

② 10 進数の 95 と 89 を加える場合，

Reg.A=95H のとき，ADD A，89H 命令を実行すると，

	1	0	0	1	0	1	0	1	Reg.A
+	1	0	0	0	1	0	0	1	n
	0	0	0	1	1	1	1	0	Reg.A

C	Z	P/V	S	N	H
1	0	1	0	0	0

次に DAA 命令を実行すると，Reg.A の下位 4 ビットが E_{16} であり，かつ C= 1 であるので，Reg.A には 66H が加えられ，結果は 84H となる(95+89=184)．

フラグ・レジスタは次のようになる．

C	Z	P/V	S	N	H
1	0	1	1	0	1

③ 10 進数の 84 から 59 を引く場合，

Reg.A=84H のとき，SUB A，59H 命令を実行すると，

1	0	0	0	0	1	0	0	Reg.A
0	1	0	1	1	0	0	1	n
0	0	1	0	1	0	1	1	Reg.A

C	Z	P/V	S	N	H
0	0	1	0	1	1

次に DAA 命令を実行すると，H= 1 でReg.A の下位 4 ビットの値が B_{16} であるので，Reg.A に FAH が加えられ，結果は 25H(84−59=25)となり，フラグ・レジスタは次のようになる．

C	Z	P/V	S	N	H
0	0	0	0	1	0

DEC (HL)

(Decrement operand (HL))　M＝3　T＝11(4,4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	1	1	0	1	0	1	35H

【命令の機能】　((HL)←(HL)−1)

レジスタ・ペアHLにより示されるメモリの内容を1だけ減じる．

【例】

① HL＝2345H，(2345H)＝37Hであるときに，この命令を実行すると，

(2345H)＝36H，

C	Z	P/V	S	N	H
●	0	0	0	1	0

② HL＝2345H，(2345H)＝80Hであるときに，この命令を実行すると，

(2345H)＝7FH，

C	Z	P/V	S	N	H
●	0	1	0	1	1

③ HL＝2345H，(2345H)＝01Hであるときに，この命令を実行すると，

(2345H)＝00H，

C	Z	P/V	S	N	H
●	1	0	0	1	0

DEC (IX+d)

(Decrement operand (IX+d))　M＝6　T＝23(4,4,3,5,4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
0	0	1	1	0	1	0	1	35H
←—			d				—→	

【命令の機能】　((IX+d)←(IX+d)−1)

インデックス・レジスタIXの内容と，ディスプレースメントdの和により示されるメモリの内容を，1だけ減じる．

【例】

① IX＝2345H，d＝67H，(23ACH)＝37Hであるときに，この命令を実行すると，

(23ACH)＝36H，

C	Z	P/V	S	N	H
●	0	0	0	1	0

② IX＝2345H，d＝C8H，(230DH)＝80Hであるときに，この命令を実行すると，

(230DH)＝7FH

C	Z	P/V	S	N	H
●	0	1	0	1	1

③ IX＝FFF0H，d＝40H，(0030H)＝01Hであるときに，この命令を実行すると，

(0030H)＝00H，

C	Z	P/V	S	N	H
●	1	0	0	1	0

DEC (IY+d)

(Decrement operand (IY+d))

M=6
T=23(4,4,3,5,4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
0	0	1	1	0	1	0	1	35H
←			d				→	

【命令の機能】 ((IY+d)←(IY+d)−1)

インデックス・レジスタIYの内容と，ディスプレースメントdの和で示される，メモリの内容を1だけ減じる．

【例】

① IY=2345H，d=67H，(23ACH)=37Hであるときに，この命令を実行すると，
(23ACH)=36H，

C	Z	P/V	S	N	H
●	0	0	0	1	0

② IY=2345H，d=C8H，(230DH)=80Hであるときに，この命令を実行すると，
(230DH)=7FH，

C	Z	P/V	S	N	H
●	0	1	0	1	1

③ IY=FFF0H，d=40H，(0030H)=01Hであるときに，この命令を実行すると，
(0030H)=00H，

C	Z	P/V	S	N	H
●	1	0	0	1	0

DEC r

(Decrement register r)

M=1
T=4

【オブジェクト・コード】

7	6	5	4	3	2	1	0
0	0	←	r	→	1	0	1

r	rの値	ニーモニック	オブジェクト・コード
B	000	DEC B	05H
C	001	DEC C	0DH
D	010	DEC D	15H
E	011	DEC E	1DH
H	100	DEC H	25H
L	101	DEC L	2DH
A	111	DEC A	3DH

【命令の機能】 (r←r−1)

レジスタrから1を減じる．

【例】

① Reg.B=01Hのとき，DEC B命令を実行すると，
Reg.B=00H，

C	Z	P/V	S	N	H
●	1	0	0	1	0

② Reg.H=80Hのとき，DEC H命令を実行すると，
Reg.H=7FH，

C	Z	P/V	S	N	H
●	0	1	0	1	1

③ Reg.A=37Hのとき，DEC A命令を実行すると，
Reg.A=36H，

C	Z	P/V	S	N	H
●	0	0	0	1	0

DEC ss

(Decrement register pair ss)

M=1
T=6

【オブジェクト・コード】

7	6	5	4	3	2	1	0
0	0	←	ss →	1	0	1	1

ss	ssの値	ニーモニック	オブジェクト・コード
BC	00	DEC BC	0BH
DE	01	DEC DE	1BH
HL	10	DEC HL	2BH
SP	11	DEC SP	3BH

【命令の機能】 (ss←ss−1)

レジスタ・ペアss(BC，DE，HL，SP)の内容を1だけ減じる．

フラグ・レジスタの状態は変化しない．

【例】

① BC=1000Hのとき，DEC BC命令を実行すると，BC=0FFFHとなる．

② HL=0001Hのとき，DEC HL命令を実行すると，HL=0000Hとなる．

DEC IX

(Decrement index register IX)

M＝2
T＝10(4,6)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
0	0	1	0	1	0	1	1	2BH

【命令の機能】　(IX←IX－1)

インデックス・レジスタIXの内容を1だけ減じる．フラグ・レジスタの状態は変化しない．

【例】

IX＝1234Hのとき，この命令を実行すると，
IX＝1233Hとなる．

DEC IY

(Decrement index register IY)

M＝2
T＝10(4,6)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
0	0	1	0	1	0	1	1	2BH

【命令の機能】　(IY←IY－1)

インデックス・レジスタIYの内容を1だけ減じる．フラグ・レジスタの状態は変化しない．

【例】

IY＝2345Hのときに，この命令を実行すると，
IY＝2344Hとなる．

DI

(Disable Interrupts)

M＝1
T＝4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	0	0	1	1	F3H

【命令の機能】　(IFF 1←0，IFF 2←0)

CPU内部のインタラプト・イネーブル・フリップフロップ(IFF 1，IFF 2)をリセットし，割り込みを禁止する．

フラグ・レジスタの状態は変化しない．

DJNZ

$B \neq 0 \begin{pmatrix} M=3 \\ T=13(5,3,5) \end{pmatrix}$　$B=0 \begin{pmatrix} M=2 \\ T=8(5,3) \end{pmatrix}$

(Decrement register B & Jump relative if register B is Not Zero)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	0	1	0	0	0	0	10H
← e−2 →								

【命令の機能】

レジスタBの内容を1だけ減じ，レジスタBの値がゼロであれば次の命令に進み，ゼロでない場合にはこの命令のアドレス＋(e−2)へ相対ジャンプする．この命令のアドレス−126～＋129バイトの範囲をジャンプすることができる．

DJNZ命令が2345H番地にあり，ジャンプ先が2389H番地である場合，(e−2)＝2389H−2345H−2＝43Hとなる．

また，DJNZ命令が2345H番地にあり，ジャンプ先が2300H番地である場合，2300H−2345H−2＝FFB9Hであるので，(e−2)＝B9Hとなる．

【例】

① 整数，10，9，8，・・・，2，1の和を求める場合．

```
        LD     B, 10
        XOR    A
LOOP:   ADD    A, B
        DJNZ   LOOP
```

この場合，DJNZ命令の(e−2)部分の値はFDHである．

② タイミング・ループ

```
        LD     B, 00H
        DJNZ   $
```

DJNZ命令を256回繰り返してから，次の命令に抜ける．DJNZ命令の(e−2)部分の値はFEHである．

EI

(Enable Interrupts)　　M＝1　T＝4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	0	1	1	FBH

【命令の機能】　　(IFF1←1，IFF2←1)

インタラプト・イネーブル・フリップフロップ(IFF1，IFF2)をいずれもセットし，割り込みを許可する．実際に割り込みが許可されるのは，この命令の次の命令の実行終了後である．

したがって，次のようにEI命令とDI命令を連続して実行しても，両命令の間で割り込みが発生することはない．

```
EI
   ←ここで割り込みは発生しない
DI
```

フラグ・レジスタの状態は変化しない．

EX (SP), HL (Exchange the location (SP) and HL)

M＝5
T＝19(4,3,4,3,5)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	0	0	1	1	E3H

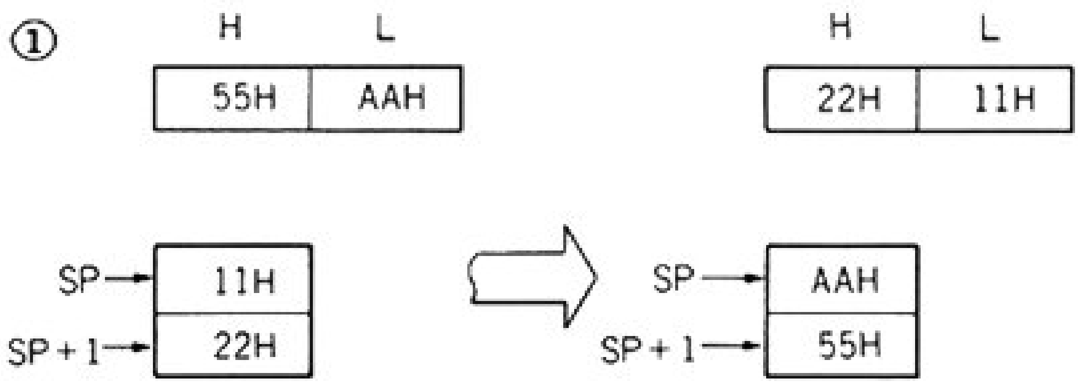
【命令の機能】　(L←→(SP)，H←→(SP+1))

レジスタLとSPが示すメモリの内容が交換され，レジスタHとSP+1が示すメモリの内容が交換される．

フラグ・レジスタの状態は変化しない．

【例】

①



② CALL 命令の後にパラメータ列があるとき．

```
CALL  SUB        SUB:  EX(SP), HL
  ⋮                      ⋮
パラメータ列             INC   HL
次の命令                  ⋮
                         INC   HL
                          ⋮
                         EX(SP), HL
                         RET
```

EX (SP), IX (Exchange the location (SP) and IX)

M＝6
T＝23(4,4,3,4,3,5)

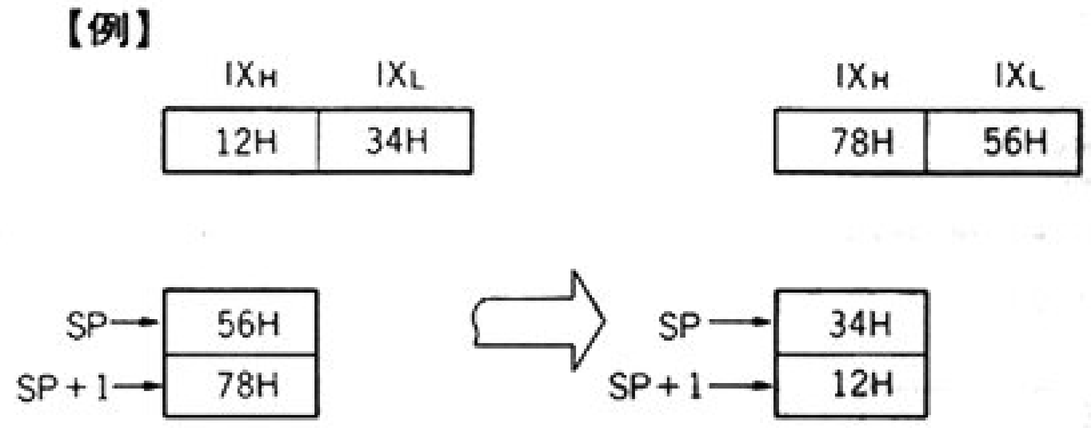
【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	1	0	0	0	1	1	E3H

【命令の機能】　(IX_L←→(SP)，IX_H←→(SP+1))

IXの下位バイト(IX_L)とSPが示すメモリの内容を交換し，IXの上位バイト(IX_H)とSP+1が示すメモリの内容を交換する．フラグ・レジスタの状態は変化しない．

【例】



EX (SP), IY (Exchange the location (SP) and IY)

M＝6
T＝23(4,4,3,4,3,5)

【オブジェクト・コード】

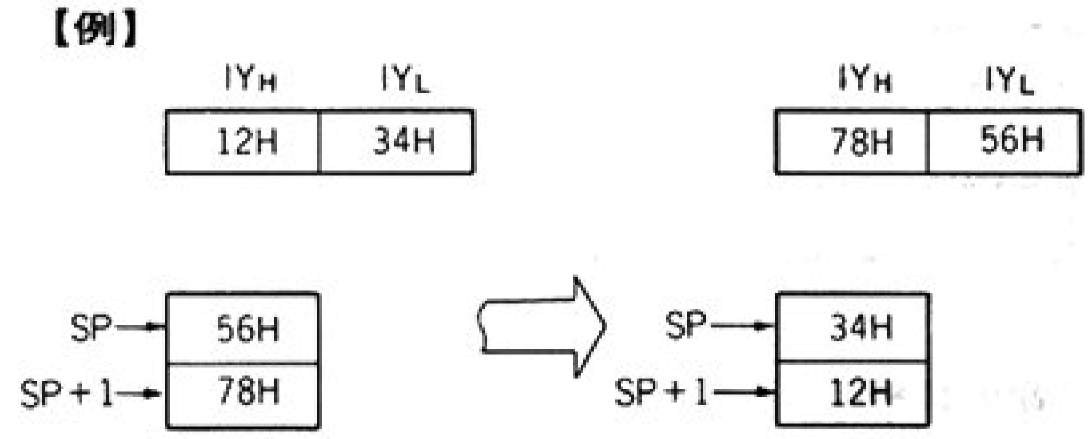
7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	1	0	0	0	1	1	E3H

【命令の機能】　(IY_L←→(SP)，IY_H←→(SP+1))

IYの下位バイト(IY_L)とSPが示すメモリの内容を交換し，IYの上位バイト(IY_H)とSP+1が示すメモリの内容を交換する．

フラグ・レジスタの状態は変化しない．

【例】



EX AF, AF'

(Exchange the contents of AF and AF') M=1 T=4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	0	0	1	0	0	0	08H

【命令の機能】 (AF←→AF')

アキュムレータ(A)とフラグ・レジスタ(F)の表と裏を交換する．フラグ・レジスタの内容は交換されるが，変化はしない．

【例】

A=37H，F=20H，A'=00H，F'=44Hのとき，この命令を実行すると，

A=00H，F=44H，A'=37H，F'=20Hとなる．

EX DE, HL

(Exchange the contents of DE and HL) M=1 T=4

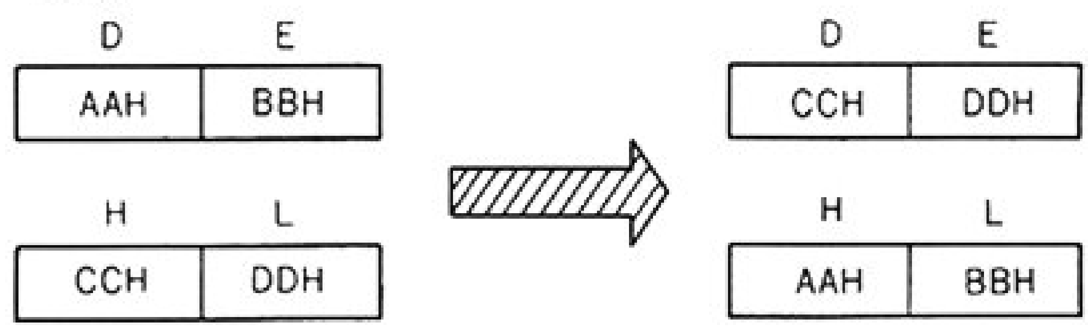
【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	0	1	1	EBH

【命令の機能】 (D←→H，E←→L)

レジスタ・ペアDEとHLの内容を変換する．つまり，レジスタDとHの内容を変換し，レジスタEとLの内容を交換する．フラグ・レジスタの状態は変化しない．

【例】



EXX

(Exchange the register bank) M=1 T=4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	0	0	1	D9H

【命令の機能】 (BC←→BC', DE←→DE', HL←→HL')

レジスタB, C, D, E, H, LとB', C', D', E', H', L'を交換する．フラグ・レジスタの状態は変化しない．

【例】

BC=1234H，DE=5678H，HL=9ABCH

BC'=FEDCH，DE'=BA98H，HL'=7654Hのとき，この命令を実行すると，

BC=FEDCH，DE=BA98H，HL=7654H

BC'=1234H，DE'=5678H，HL'=9ABCHとなる．

HALT

(Halt CPU—Wait for interrupt or reset) M=1 T=4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	1	1	1	0	1	1	0	76H

【命令の機能】

この命令を実行すると，CPUは割り込みを受け付けるか，リセット信号が印加されるまで，動作を停止する(HALT状態)．

HALT状態にあるときには，CPUはダイナミックRAMのリフレッシュ動作を維持するために，NOP命令を反復する．

IM 0

(Set interrupt mode 0) M=2 T=8

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	0	0	0	1	1	0	46H

【命令の機能】

割り込みモード0にセットする．フラグ・レジスタの状態は変化しない．

IM 1

(Set interrupt mode 1) M=2 T=8

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	0	1	0	1	1	0	56H

【命令の機能】

割り込みモード1にセットする．フラグ・レジスタの状態は変化しない．

IM 2

(Set interrupt mode 2) M=2 T=8

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	0	1	1	1	1	0	5EH

【命令の機能】

割り込みモード2にセットする．フラグ・レジスタの状態は変化しない．

IN A, (n)

(Load the Acc. with input from device n) M=3 T=11(4, 3, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	0	1	1	DBH
←			n				→	

【例】

I/Oポート55HのデータがAAHであるとき，
IN A, (55H)命令を実行すると，
Reg.A=AAHとなる．

【命令の機能】 (A←(n))

I/Oポートnのデータをアキュムレータに読み取る．
フラグ・レジスタの状態は変化しない．

IN r, (C)

(Load the Reg. r with input from device (C))　M=3　T=12(4, 4, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	←—	r	—→	0	0	0	

r	rの値	ニーモニック	オブジェクト・コード
B	000	IN B, (c)	40H
C	001	IN C, (c)	48H
D	010	IN D, (c)	50H
E	011	IN E, (c)	58H
H	100	IN H, (c)	60H
L	101	IN L, (c)	68H
A	111	IN A, (c)	78H

【命令の機能】　(r←(C))

レジスタCの内容により示されるI/Oポートのデータを、レジスタrに読み取る。

【例】

① Reg.C=55H, I/Oポート55HのデータがA1Hであるときに、IN H, (c)命令を実行すると、
Reg.H=A1H,

C	Z	P/V	S	N	H
●	0	0	1	0	0

② Reg.C=55H, I/Oポート55Hのデータが00Hであるときに、IN C, (c)命令を実行すると、
Reg.C=00H,

C	Z	P/V	S	N	H
●	1	1	0	0	0

INC (HL)

(Increment Location(HL))　M=3　T=11(4, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	1	1	0	1	0	0	34H

【命令の機能】　((HL)←(HL)+1)

レジスタ・ペアHLで示されるメモリの内容に1を加える。

【例】

① HL=2345H, (2345H)=00Hであるときに、この命令を実行すると、
(2345H)=01H,

C	Z	P/V	S	N	H
●	0	0	0	0	0

② HL=2345H, (2345H)=7FHであるときに、この命令を実行すると、
(2345H)=80H,

C	Z	P/V	S	N	H
●	0	1	1	0	1

③ HL=2345H, (2345H)=FFHであるときに、この命令を実行すると、
(2345H)=00H,

C	Z	P/V	S	N	H
●	1	0	0	0	1

INC (IX+d)

(Increment Location(IX+d))

M=6
T=23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
0	0	1	1	0	1	0	0	34H
←d→								

【命令の機能】 ((IX+d)←(IX+d)+1)

インデックス・レジスタ IX の内容にディスプレースメント d を加えた値が示すメモリの内容に 1 を加える．

【例】

① IX=2345H, d=67H, (23ACH)=00H であるときに，この命令を実行すると，

(23ACH)=01H,

C	Z	P/V	S	N	H
●	0	0	0	0	0

② IX=2345H, d=ABH, (22F0H)=7FH であるときに，この命令を実行すると，

(22F0H)=80H,

C	Z	P/V	S	N	H
●	0	1	1	0	1

③ IX=2345H, d=FFH, (2344H)=FFH であるときに，この命令を実行すると，

(2344H)=00H,

C	Z	P/V	S	N	H
●	1	0	0	0	1

INC (IY+d)

(Increment Location(IY+d))

M=6
T=23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
0	0	1	1	0	1	0	0	34H
←d→								

【命令の機能】 ((IY+d)←(IY+d)+1)

インデックス・レジスタ IY の内容に，ディスプレースメント d を加えた値が示すメモリの内容に，1 を加える．

【例】

① IY=2345H, d=67H, (23ACH)=00H であるときに，この命令を実行すると，

(23ACH)=01H,

C	Z	P/V	S	N	H
●	0	0	0	0	0

② IY=2345H, d=ABH, (22F0H)=7FH であるときに，この命令を実行すると，

(22F0H)=80H,

C	Z	P/V	S	N	H
●	0	1	1	0	1

③ IY=2345H, d=FFH, (2344H)=FFH であるときに，この命令を実行すると，

(2344H)=00H

C	Z	P/V	S	N	H
●	1	0	0	0	1

INC r

(Increment Reg. r)　M=1 T=4

【オブジェクト・コード】

7	6	5	4	3	2	1	0
0	0	←──	r	──→	1	0	0

r	rの値	ニーモニック	オブジェクト・コード
B	000	INC B	04H
C	001	INC C	0CH
D	010	INC D	14H
E	011	INC E	1CH
H	100	INC H	24H
L	101	INC L	2CH
A	111	INC A	3CH

【命令の機能】　(r←r+1)

レジスタ r の内容に 1 を加える．

【例】

① Reg.B=00H であるときに，INC B 命令を実行すると，

Reg.B=01H,

C	Z	P/V	S	N	H
●	0	0	0	0	0

② Reg.D=7FH であるときに，INC D 命令を実行すると，

Reg.D=80H,

C	Z	P/V	S	N	H
●	0	1	1	0	1

③ Reg.A=FFH であるときに，INC A 命令を実行すると，

Reg.A=00H,

C	Z	P/V	S	N	H
●	1	0	0	0	1

INC ss

(Increment Register pair ss)　M=1 T=6

【オブジェクト・コード】

7	6	5	4	3	2	1	0
0	0	s	s	0	0	1	1

ss	ssの値	ニーモニック	オブジェクト・コード
BC	00	INC BC	03H
DE	01	INC DE	13H
HL	10	INC HL	23H
SP	11	INC SP	33H

【命令の機能】　(ss←ss+1)

レジスタ・ペア ss に 1 を加える．フラグ・レジスタの状態は変化しない．

【例】

① BC=00FFH であるときに、INC BC 命令を実行すると，

BC=0100H となる．

② SP=7FFFH であるときに，INC SP 命令を実行すると，

SP=8000H となる．

INC IX

(Increment Index Reg. IX)　M=2 T=10(4, 6)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
0	0	1	0	0	0	1	1	23H

【命令の機能】　(IX←IX+1)

インデックス・レジスタ IX の内容に 1 を加える．フラグ・レジスタの状態は変化しない．

【例】

IX=7FFFH であるときに，この命令を実行すると，

IX=8000H となる．

INC IY

(Increment Index Reg. IY)　M=2　T=10(4, 6)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
0	0	1	0	0	0	1	1	23H

【命令の機能】　(IY←IY+1)

インデックス・レジスタIYの内容に1を加える。

【例】

IY=7FFFHであるときに、この命令を実行すると、
IY=8000Hとなる。

IND

(Load location(HL)with input from port(C), decrement HL and B)　M=4　T=16(4, 5, 3, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	0	1	0	1	0	AAH

【命令の機能　((HL)←(C), B←B−1, HL←HL−1)

レジスタCで示されるI/Oポートのデータを、レジスタ・ペアHLで示されるメモリにストアする。その後、レジスタBおよびレジスタ・ペアHLを1だけ減じる。

【例】

① Reg.B=55H, Reg.C=AAH, HL=2345Hであるときに、この命令を実行すると、
(HL)=ポートAAHのデータ, Reg.B=54H, HL=2344H,

C	Z	P/V	S	N	H
●	0	×	×	1	×

② Reg.B=01H, Reg.C=AAH, HL=2345Hであるときに、この命令を実行すると、
(HL)=ポートAAHのデータ, Reg.B=00H, HL=2344H,

C	Z	P/V	S	N	H
●	1	×	×	1	×

INDR

B≠0 (M=5, T=21(4, 5, 3, 4, 5)), B=0 (M=4, T=16(4, 5, 3, 4))

(Load location(HL)with input from port(C), decrement HL and B, repeat until B=0.)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	1	1	0	1	0	BAH

【命令の機能】((HL)←(C), B←B−1, HL←HL−1)

レジスタCで示されるI/Oポートのデータを、レジスタ・ペアHLで示されるメモリにストアする。その後レジスタB、およびレジスタ・ペアHLを1だけ減じる。

以上の過程はレジスタB=0となるまで繰り返される。

【例】

Reg.B=00H, Reg.C=55H, HL=24FFHであり、I/Oポート55Hからデータ00H〜FFHが順に読み出されるものとし、この命令を実行すると、
Reg.B=00H,
HL=23FFH,

C	Z	P/V	S	N	H
●	1	×	×	1	×

24FFH	24FEH	24FDH	24FCH	〜	24F3H	24F2H	24F1H	24F0H
00H	01H	02H	03H	〜	FCH	FDH	FEH	FFH

INI

(Load location(HL)with input from(C), increment HL and decrement B)

M＝4
T＝16(4, 5, 3, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	0	0	0	1	0	A2H

【命令の機能】 ((HL)←(C), HL←HL＋1, B←B－1)

レジスタCで示されるI/Oポートのデータを，レジスタ・ペアHLで示されるメモリにストアする．その後，レジスタ・ペアHLに1を加え，レジスタBから1を減じる．

【例】

① Reg.B＝55H, Reg.C＝AAH, HL＝2345Hであるときに，この命令を実行すると，
(2345H)＝ポートAAHのデータ，Reg.B＝54H,
HL＝2346H,

C	Z	P/V	S	N	H
●	0	×	×	1	×

② Reg.B＝01H, Reg.C＝AAH, HL＝2345Hであるときに，この命令を実行すると，
(2345H)＝ポートAAHのデータ，Reg.B＝00H,
HL＝2346H,

C	Z	P/V	S	N	H
●	1	×	×	1	×

INIR

B≠0 (M＝5, T＝21(4, 5, 3, 4, 5)), B＝0 (M＝4, T＝16(4, 5, 3, 4))

(Load location(HL)with input from port(C), increment HL and decrement B, repeat until B＝0)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	1	0	0	1	0	B2H

【命令の機能】 ((HL)←(C), B←B－1, HL←HL＋1)

レジスタCで示されるI/Oポートのデータを，レジスタ・ペアHLで示されるメモリにストアする．その後レジスタBから1を減じ，レジスタ・ペアHLに1を加える．

以上の過程はレジスタB＝0になるまで繰り返される．

【例】

Reg.C＝55H, Reg.B＝10H, HL＝2345Hであり，I/Oポート55Hからデータ20H～2FHが順次読み出されるものとしたときに，この命令を実行すると，
Reg.B＝00H,
HL＝2355H,

C	Z	P/V	S	N	H
●	1	×	×	1	×

2345H	2346H	2347H		2351H	2352H	2353H	2354H
20H	21H	22H	～	2CH	2DH	2EH	2FH

JP nn

(Unconditional jump to location nn)

M＝3
T＝10(4, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	0	0	1	1	C3H
←			n				→	
←			n				→	

【命令の機能】 (PC←nn)

命令のオペランドnnがPCにロードされる．したがってnn番地へジャンプすることになる．フラグ・レジスタの状態は変化しない．

【例】

nn＝2345Hであるときに，JP 2345H命令(C3H, 45H, 23H)を実行すると，
2345H番地へジャンプする．

JP (HL)

(Unconditional jump to (HL))　M=1 T=4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	0	0	1	E9H

【命令の機能】　(PC←HL)

レジスタ・ペアHLの内容をPCにロードする．したがってレジスタ・ペアHLが示すアドレスへジャンプする．

フラグ・レジスタの状態は変化しない．

【例】

HL＝2345Hであるときに，この命令を実行すると，2345H番地へジャンプする．

JP (IX)

(Unconditional jump to (IX))　M=2 T=8(4, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	1	0	1	0	0	1	E9H

【命令の機能】　(PC←IX)

インデックス・レジスタIXの内容をPCにロードする．したがって，IXが示すアドレスへジャンプする．

【例】

IX＝2345Hであるときに，この命令を実行すると，2345H番地へジャンプする．

JP (IY)

(Unconditional jump to (IY))　M=2 T=8(4, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	1	0	1	0	0	1	E9H

【命令の機能】　(PC←IY)

インデックス・レジスタIYの内容をPCにロードする．したがってIYが示すアドレスへジャンプする．

フラグ・レジスタの値は変化しない．

【例】

IY＝2345Hであるときに，この命令を実行すると，2345H番地へジャンプする．

JP　cc, nn

(Jump on condition)　M＝3　T＝10(4, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0
1	1	c	c	c	0	1	0
←n→							
←n→							

cc	ccの意味	ccの値	ニーモニック	オブジェクト・コード
NZ	Non Zero	0 0 0	JP NZ, nn	C 2 H
Z	Zero	0 0 1	JP Z, nn	C A H
NC	No Carry	0 1 0	JP NC, nn	D 2 H
C	Carry	0 1 1	JP C, nn	D A H
PO	Parity Odd	1 0 0	JP PO, nn	E 2 H
PE	Parity Even	1 0 1	JP PE, nn	E A H
P	Positive	1 1 0	JP P, nn	F 2 H
M	Minus	1 1 1	JP M, nn	F A H

【命令の機能】　(If cc is true, PC←nn)

判定条件(cc)が真(true)であるときには，アドレスnnへジャンプする．判定条件が真でないときには次の命令に進む．

フラグ・レジスタの状態は変化しない．

【例】

① Zフラグが“1”であるときに，JP Z, 1234H命令を実行すると，1234H番地へジャンプする．

② Sフラグが“0”であるときに，1000H番地にあるJP M, 1234H命令を実行すると，JP命令の次の1003H番地の命令を実行する．

JR　cc, e

(Jump Relative on condition cc)　ジャンプ時(M＝3　T＝12(4, 3, 5))，非ジャンプ時(M＝2　T＝7(4, 3))

【オブジェクト・コード】

7	6	5	4	3	2	1	0
0	0	1	c	c	0	0	0
←e−2→							

cc	ccの意味	ccの値	ニーモニック	オブジェクト・コード
NZ	Non Zero	0 0	JR NZ, e	2 0 H
Z	Zero	0 1	JR Z, e	2 8 H
NC	Non Carry	1 0	JR NC, e	3 0 H
C	Carry	1 1	JR C, e	3 8 H

【命令の機能】　(If cc is true, PC←PC＋e)

判定条件が真(true)であるときには，この命令のアドレス＋e番地へジャンプする．判定条件が真でないときには，次の命令を実行する．

フラグ・レジスタの状態は変化しない．

【例】

①

アドレス	ラベル	命令
1234H		JR　Z, DEST
⋮		
1287H	DEST：	

Zフラグが“1”であるときに，上記のJR　Z, DEST命令を実行すると，1287H番地へジャンプする．このJR　Z, DEST命令のオブジェクト・コードは，

[28H | 51H]　(1287H－1234H－2＝0051H)

②

アドレス	ラベル	命令
1234H	DEST：	
⋮		
1287H		JR　NC, DEST

Cフラグが“0”であるときに，上記のJR　NC, DEST命令を実行すると，1234H番地へジャンプする．このJR　NC, DEST命令のオブジェクト・コードは，

[30H | ABH]　(1234H－1287H－2＝FFABH)

JR e

(Jump Relative unconditionally)　M=3　T=12(4, 3, 5)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	0	1	1	0	0	0	18H
← e − 2 →								

【命令の機能】　(PC←PC+e)

この命令のある番地 +e 番地へジャンプする.

フラグ・レジスタの状態は変化しない.

【例】

①

アドレス	ラベル	命　令
1234H		JR　DEST
⋮		
12A6H	DEST :	

上記の JR　DEST 命令を実行すると，12A6H 番地へジャンプする.

この JR　DEST 命令のオブジェクト・コードは,

18H | 70H　(12A6H−1234H−2=0070H)

②

アドレス	ラベル	命　令
1234H	DEST :	
⋮		
12A6H		JR　DEST

上記の JR　DEST 命令を実行すると，1234H 番地へジャンプする．この JR　DEST 命令のオブジェクト・コードは,

18H | 8CH　(1234H−12A6H−2=FF8CH)

③ その命令自体のアドレスへジャンプする(無限ループ)JR　$ 命令のオブジェクト・コードは,

18H | FEH　(XXXXH−XXXXH−2=FFFEH)

LD (ss), A

(Load location (ss) with Acc.)　M=2　T=7(4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0
0	0	0	ss	0	0	1	0

ss	ssの値	ニーモニック	オブジェクト・コード
BC	0	LD　(BC), A	0 2 H
DE	1	LD　(DE), A	1 2 H

【命令の機能】　((ss)←A)

レジスタ・ペア ss(BC または DE)で示されるアドレスにアキュムレータのデータをロードする．フラグ・レジスタの状態は変化しない.

【例】

① BC=2345H, Reg.A=55H であるときに,
LD　(BC), A 命令を実行すると,
(2345H)=55H となる.

② DE=ABCDH, Reg.A=AAH であるときに,
LD　(DE), A 命令を実行すると,
(ABCDH)=AAH となる.

LD (HL), r

(Load location (HL) with Reg.r)　M＝2　T＝7(4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0
0	1	1	1	0	←r→		

r	rの値	ニーモニック	オブジェクト・コード
B	0 0 0	LD (HL), B	7 0 H
C	0 0 1	LD (HL), C	7 1 H
D	0 1 0	LD (HL), D	7 2 H
E	0 1 1	LD (HL), E	7 3 H
H	1 0 0	LD (HL), H	7 4 H
L	1 0 1	LD (HL), L	7 5 H
A	1 1 1	LD (HL), A	7 7 H

【命令の機能】　((HL)←r)

レジスタrの内容を，レジスタ・ペアHLで示されるアドレスにロードする．フラグ・レジスタの状態は変化しない．

【例】

① HL＝2345H，Reg.B＝55Hであるときに，
LD (HL)，B命令を実行すると，
(2345H)＝55Hとなる．

② HL＝5678H，Reg.A＝AAHであるときに，
LD (HL)，A命令を実行すると，
(5678H)＝AAHとなる．

LD (HL), n

(Load location (HL) with n)　M＝3　T＝10(4, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	1	1	0	1	1	0	36H
←n→								

【命令の機能】　((HL)←n)

レジスタ・ペアHLにより示されるアドレスに，イミーディエイト・データnをロードする．

フラグ・レジスタの状態は変化しない．

【例】

HL＝2345Hであるときに，LD (HL)，55H命令を実行すると，
(2345H)＝55Hとなる．

LD (IX+d), r

(Load location (IX+d) with r)　M＝5　T＝19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
0	1	1	1	0	r			
←d→								

r	rの値	ニーモニック	オブジェクト・コード
B	0 0 0	LD (IX+d), B	7 0 H
C	0 0 1	LD (IX+d), C	7 1 H
D	0 1 0	LD (IX+d), D	7 2 H
E	0 1 1	LD (IX+d), E	7 3 H
H	1 0 0	LD (IX+d), H	7 4 H
L	1 0 1	LD (IX+d), L	7 5 H
A	1 1 1	LD (IX+d), A	7 7 H

【命令の機能】　((IX＋d)←r)

インデックス・レジスタIXとディスプレースメントdの和で示されるアドレスに，レジスタrの内容をロードする．

フラグ・レジスタの状態は変化しない．

【例】

① IX＝2345H，d＝67H，Reg.B＝AAHであるときに，
LD (IX＋d)，B命令を実行すると，
(23ACH)＝AAHとなる．

② IX＝2345H，d＝C6H，Reg.A＝55H であるときに，
LD (IX＋d)，A命令を実行すると，
(230BH)＝55Hとなる．

LD (IX+d), n

(Load location (IX+d) with n)

M＝5
T＝19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
0	0	1	1	0	1	1	0	36H
←d→								
←n→								

【命令の機能】　((IX＋d)←n)

インデックス・レジスタIXとディスプレースメントdの和により示されるアドレスに、イミーディエイト・データnをロードする。

フラグ・レジスタの状態は変化しない。

【例】

① IX＝2345H, d＝67H, n＝AAHであるときに、この命令を実行すると、
(23ACH)＝AAHとなる。

② IX＝2345H, d＝C6H, n＝55Hであるときに、この命令を実行すると、
(230BH)＝55Hとなる。

LD (IY+d), r

(Load location (IY+d) with r)

M＝5
T＝19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
0	1	1	1	0	←r→			
←d→								

r	rの値	ニーモニック	オブジェクト・コード
B	0 0 0	LD (IY＋d), B	7 0 H
C	0 0 1	LD (IY＋d), C	7 1 H
D	0 1 0	LD (IY＋d), D	7 2 H
E	0 1 1	LD (IY＋d), E	7 3 H
H	1 0 0	LD (IY＋d), H	7 4 H
L	1 0 1	LD (IY＋d), L	7 5 H
A	1 1 1	LD (IY＋d), A	7 7 H

【命令の機能】　((IY＋d)←r)

インデックス・レジスタIYとディスプレースメントdの和により示されるアドレスに、レジスタrの内容をロードする。

フラグ・レジスタの状態は変化しない。

【例】

① IY＝2345H, d＝67H, Reg.B＝55Hのときに、
LD (IY＋d), B命令を実行すると、
(23ACH)＝55Hとなる。

② IY＝2345H, d＝C6H, Reg.A＝AAHのときに、
LD (IY＋d), A命令を実行すると、
(230BH)＝AAHとなる。

LD (IY+d), n

(Load location (IY+d) with n)

M＝5
T＝19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
0	0	1	1	0	1	1	0	36H
←d→								
←n→								

【命令の機能】　((IY＋d)←n)

インデックス・レジスタIYとディスプレースメントdの和で示されるアドレスに、イミーディエイト・データnをロードする。

フラグ・レジスタの状態は変化しない。

【例】

① IX＝2345H, d＝67H, n＝55Hのとき、この命令を実行すると、
(23ACH)＝55Hとなる。

② IX＝2345H, d＝C6H, n＝AAHのとき、この命令を実行すると、
(230BH)＝AAHとなる。

LD (nn), A

(Load location (nn) with Acc.)　M＝4　T＝13(4, 3, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	1	1	0	0	1	0	32H
←			n				→	
←			n				→	

【命令の機能】　((nn)←A)

アキュムレータの内容を nn により示されるアドレスのメモリにロードする。

フラグ・レジスタの状態は変化しない。

【例】

Reg.A＝55H, nn＝2345H のとき，この命令を実行すると，

(2345H)＝55H となる。

LD (nn), ss

(Load location (nn) with ss)　M＝6　T＝20(4, 4, 3, 3, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	s	s	0	0	1	1	
←			n				→	
←			n				→	

ss	ssの値	ニーモニック	オブジェクト・コード
BC	0 0	LD (nn), BC	4 3 H
DE	0 1	LD (nn), DE	5 3 H
HL	1 0	LD (nn), HL	6 3 H
SP	1 1	LD (nn), SP	7 3 H

【命令の機能】　((nn)←ss_L, (nn＋1)←ss_H)

nn で示されるアドレスに，レジスタ・ペア ss の内容をロードする。ss の下位バイトが nn 番地に，ssの上位バイトが nn＋1 番地にロードされる。

フラグ・レジスタの状態は変化しない。

【例】

SP＝1234H, nn＝2345H のとき，LD (nn), SP 命令を実行すると，

(2345H)＝34H, (2346H)＝12H となる。

LD (nn), HL

(Load location (nn) with HL)　M＝5　T＝16(4, 3, 3, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	1	0	0	0	1	0	22H
←			n				→	
←			n				→	

【命令の機能】　((nn)←L, (nn＋1)←H)

レジスタ・ペア HL の内容を，nn で示されるアドレスにロードする。nn 番地にはレジスタ L がロードされ，nn＋1 番地には，レジスタ H がロードされる。

フラグ・レジスタの状態は変化しない。

【例】

HL＝1234H, nn＝2345H のとき，この命令を実行すると，

(2345H)＝34H, (2346H)＝12H となる。

LD (nn), IX

(Load location (nn) with IX)　M＝6　T＝20(4, 4, 3, 3, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
0	0	1	0	0	0	1	0	22H
←			n				→	
←			n				→	

【命令の機能】　((nn)←IX_L, (nn＋1)←IX_H)

インデックス・レジスタ IX の内容を nn で示されるアドレスにロードする．IX の下位バイトは nn 番地にロードされ，IX の上位バイトは nn＋1 番地にロードされる．

フラグ・レジスタの状態は変化しない．

【例】

IX＝2345H，nn＝6789H であるときに，この命令を実行すると，

(6789H)＝45H，(678AH)＝23H となる．

LD (nn), IY

(Load location (nn) with IY)　M＝6　T＝20(4, 4, 3, 3, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
0	0	1	0	0	0	1	0	22H
←			n				→	
←			n				→	

【命令の機能】　((nn)←IY_L, (nn＋1)←IY_H)

インデックス・レジスタ IY の内容を，nn で示されるアドレスにロードする．IY の下位バイトは nn 番地にロードされ，IY の上位バイトは nn＋1 番地にロードされる．

フラグ・レジスタの状態は変化しない．

【例】

IY＝2345H，nn＝6789H のとき，この命令を実行すると，

(6789H)＝45H，(678AH)＝23H となる．

LD A, (ss)

(Load Acc. with location (ss))　M＝2　T＝7(4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0
0	0	0	ss	1	0	1	0

ss	ssの値	ニーモニック	オブジェクト・コード
BC	0	LD A, (BC)	0 A H
DE	1	LD A, (DE)	1 A H

【命令の機能】　(A←(ss))

レジスタ・ペア ss(BC または DE)が示すアドレスのデータを，アキュムレータにロードする．

フラグ・レジスタの状態は変化しない．

【例】

① BC＝2345H，(2345H)＝55H であるときに，
LD A, (BC)命令を実行すると，
Reg.A＝55H となる．

② DE＝AABBH，(AABBH)＝AAH であるときに，
LD A, (DE)命令を実行すると，
Reg.A＝AAH となる．

LD r, (HL)

(Load Reg.r with location (HL)) M=2 T=7(4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0
0	1	←	r	→	1	1	0

r	rの値	ニーモニック	オブジェクト・コード
B	0 0 0	LD B, (HL)	4 6 H
C	0 0 1	LD C, (HL)	4 E H
D	0 1 0	LD D, (HL)	5 6 H
E	0 1 1	LD E, (HL)	5 E H
H	1 0 0	LD H, (HL)	6 6 H
L	1 0 1	LD L, (HL)	6 E H
A	1 1 1	LD A, (HL)	7 E H

【命令の機能】 (r←(HL))

レジスタ・ペアHLが示すアドレスのデータをレジスタrにロードする．フラグ・レジスタの状態は変化しない．

【例】

① HL＝2345H，(2345H)＝55Hであるときに，LD B，(HL)命令を実行すると，Reg.B＝55Hとなる．

② HL＝AABBH，(AABBH)＝DEHであるときに，LD H，(HL)命令を実行すると，Reg.H＝DEHとなる．

LD r, (IX+d)

(Load Reg.r with location (IX+d)) M=5 T=19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
0	1	←	r	→	1	1	0	
←			d				→	

r	rの値	ニーモニック	オブジェクト・コード
B	0 0 0	LD B, (IX+d)	4 6 H
C	0 0 1	LD C, (IX+d)	4 E H
D	0 1 0	LD D, (IX+d)	5 6 H
E	0 1 1	LD E, (IX+d)	5 E H
H	1 0 0	LD H, (IX+d)	6 6 H
L	1 0 1	LD L, (IX+d)	6 E H
A	1 1 1	LD A, (IX+d)	7 E H

【命令の機能】 (r←(IX＋d))

インデックス・レジスタIXとディスプレースメントdの和で示されるアドレスのデータを，レジスタrにロードする．

フラグ・レジスタは変化しない．

【例】

① IX＝2345H，d＝67H，(23ACH)＝55Hであるときに，LD B，(IX＋d)命令を実行すると，Reg.B＝55Hとなる．

② IX＝2345H，d＝C6H，(230BH)＝AAHであるときに，LD L，(IX＋d)命令を実行すると，Reg.L＝AAHとなる．

LD r, (IY+d) (Load Reg. r with location (IY+d))

M=5
T=19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
0	1	←	r	→	1	1	0	
←			d				→	

r	rの値	ニーモニック	オブジェクト・コード
B	0 0 0	LD B, (IY+d)	4 6 H
C	0 0 1	LD C, (IY+d)	4 E H
D	0 1 0	LD D, (IY+d)	5 6 H
E	0 1 1	LD E, (IY+d)	5 E H
H	1 0 0	LD H, (IY+d)	6 6 H
L	1 0 1	LD L, (IY+d)	6 E H
A	1 1 1	LD A, (IY+d)	7 E H

【命令の機能】

インデックス・レジスタIYとディスプレースメントdの和が示すアドレスのデータを，レジスタrにロードする．

フラグ・レジスタの状態は変化しない．

【例】

① IY=2345H, d=67H, (23CAH)=55Hであるときに，LD B, (IY+d)命令を実行すると，
Reg.B=55Hとなる．

② IY=2345H, d=C6H, (230BH)=AAHであるときに，LD L, (IY+d)命令を実行すると，
Reg.L=AAHとなる．

LD A, (nn) (Load Acc. with location (nn))

M=4
T=13(4, 3, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	1	1	1	0	1	0	3AH
←			n				→	
←			n				→	

【命令の機能】 (A←(nn))

nnで示されるアドレスのデータをアキュムレータにロードする．

フラグ・レジスタの状態は変化しない．

【例】

nn=2345H, (2345H)=55Hであるときに，この命令を実行すると，
Reg.A=55Hとなる．

LD r, r' (Load Reg.r with Reg.r')

M=1
T=4

【オブジェクト・コード】

7	6	5	4	3	2	1	0
0	1	←	r	→	←	r'	→

		r'						
		B	C	D	E	H	L	A
r	B	40	41	42	43	44	45	47
	C	48	49	4A	4B	4C	4D	4F
	D	50	51	52	53	54	55	57
	E	58	59	5A	5B	5C	5D	5F
	H	60	61	62	63	64	65	67
	L	68	69	6A	6B	6C	6D	6F
	A	78	79	7A	7B	7C	7D	7F

【命令の機能】 (r←r')

レジスタr'の内容をレジスタrにロードする．

フラグ・レジスタの状態は変化しない．

【例】

Reg.B=45H, Reg.A=67Hであるときに，LD B, A命令を実行すると，
Reg.B=67Hとなる．

LD r, n

(Load Reg.r with n) M＝2 T＝7(4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0
0	0	←	r	→	1	1	0
←			n				→

r	rの値	ニーモニック	オブジェクト・コード
B	000	LD B, n	06H
C	001	LD C, n	0EH
D	010	LD D, n	16H
E	011	LD E, n	1EH
H	100	LD H, n	26H
L	101	LD L, n	2EH
A	111	LD A, n	3EH

【命令の機能】 (r←n)

イミーディエイト・データnをレジスタrにロードする.

フラグ・レジスタの状態は変化しない.

【例】

① LD B, 55H命令を実行すると,
Reg.B＝55Hとなる.

② LD A, AAH命令を実行すると,
Reg.A＝AAHとなる.

LD ss, (nn)

(Load Reg.pair ss with location (nn)) M＝6 T＝20(4, 4, 3, 3, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	s	s	1	0	1	1	
←			n				→	
←			n				→	

ss	ssの値	ニーモニック	オブジェクト・コード
BC	00	LD BC, (nn)	4BH
DE	01	LD DE, (nn)	5BH
HL	10	LD HL, (nn)	6BH
SP	11	LD SP, (nn)	7BH

【命令の機能】 (ss_L←(nn), ss_H←(nn＋1))

nnで示されるアドレスのデータを，レジスタ・ペアssにロードする．フラグ・レジスタの状態は変化しない．nn番地のデータはssの下位バイトにロードされ，nn＋1番地のデータはssの上位バイトにロードされる．

【例】

(2345H)＝67H，(2346H)＝89Hであるときに，
LD BC, (2345H)命令を実行すると，
Reg.B＝89H，Reg.C＝67Hとなる．

LD ss, nn

(Load Reg.pair ss with nn)

M=3
T=10(4, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0
0	0	s	s	0	0	0	1

← n →

← n →

ss	ssの値	ニーモニック	オブジェクト・コード
BC	0 0	LD BC, nn	0 1 H
DE	0 1	LD DE, nn	1 1 H
HL	1 0	LD HL, nn	2 1 H
SP	1 1	LD SP, nn	3 1 H

【命令の機能】 (ss←nn)

2バイトのイミーディエイト・データnnをレジスタ・ペアssにロードする。

この命令の第2バイトがssの下位バイトに、第3バイトがssの上位バイトにロードされる。

【例】

① LD HL, 1234H 命令を実行すると、
Reg.H=12H, Reg.L=34H となる。

② LD SP, 789AH 命令を実行すると、
SP=789AH となる。

LD HL, (nn)

(Load HL with location (nn))

M=5
T=16(4, 3, 3, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	1	0	1	0	1	0	2AH

← n →

← n →

【命令の機能】 (H←(nn+1), L←(nn))

nnで示されるアドレスのデータを、レジスタ・ペアHLにロードする。

nn番地のデータはレジスタLに、nn+1番地のデータはレジスタHにロードされる。

【例】

(2345H)=67H, (2346H)=89Hのときに、
LD HL, (2345H)命令を実行すると、
Reg.H=89H, Reg.L=67H となる。

LD IX, (nn)

(Load IX with location (nn))

M=6
T=20(4, 4, 3, 3, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
0	0	1	0	1	0	1	0	2AH

← n →

← n →

【命令の機能】 (IX_L←(nn), IX_H←(nn+1))

nnで示されるアドレスのデータをインデックス・レジスタIXにロードする。IXの下位バイトにはnn番地、IXの上位バイトにはnn+1番地のデータがロードされる。

フラグ・レジスタの状態は変化しない。

【例】

(2345H)=67H, (2346H)=89Hであるときに、
LD IX, (2345H)命令を実行すると、
IX=8967(IX_L=67H, IX_H=89H)となる。

LD IX, nn

(Load IX with nn) M＝4 T＝14(4,4,3,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
0	0	1	0	0	0	0	1	21H
← n →								
← n →								

【命令の機能】 (IX←nn)

イミーディエイト・データ nn を IX にロードする．フラグ・レジスタの状態は変化しない．

【例】

LD IX, 2345H 命令を実行すると，
IX＝2345H (IX_L＝45H, IX_H＝23H) となる．

LD IY, (nn)

(Load IY with location (nn)) M＝6 T＝20(4,4,3,3,3,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
0	0	1	0	1	0	1	0	2AH
← n →								
← n →								

【命令の機能】 (IY_L←(nn), IY_H←(nn＋1))

nn で示されるアドレスのデータを，インデックス・レジスタ IY にロードする．IY の下位バイトには nn 番地，IY の上位バイトには nn＋1 番地のデータがロードされる．

フラグ・レジスタの状態は変化しない．

【例】

(2345H)＝67H, (2346H)＝89H であるときに，
LD IY, (2345H) 命令を実行すると，
IY＝8967H (IY_L＝67H, IY_H＝89H) となる．

LD IY, nn

(Load IY with nn) M＝4 T＝14(4,4,3,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
0	0	1	0	0	0	0	1	21H
← n →								
← n →								

【命令の機能】 (IY←nn)

イミーディエイト・データ nn を IY にロードする．フラグ・レジスタの状態は変化しない．

【例】

LD IY, 2345H 命令を実行すると，IY＝2345H (IY_L＝45H, IY_H＝23H) となる．

LD SP, HL

(Load SP with HL) M＝1 T＝6

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	0	0	1	F9H

【命令の機能】 (SP←HL)

レジスタ・ペア HL の内容を，スタック・ポインタ SP にロードする．

フラグ・レジスタの状態は変化しない．

【例】

HL＝2345H であるときに，この命令を実行すると，
SP＝2345H となる．

LD SP, IX

(Load SP with IX) M=2 T=10(4, 6)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	1	1	1	0	0	1	F9H

【命令の機能】 (SP←IX)

インデックス・レジスタIXの内容をスタック・ポインタSPにロードする．

フラグ・レジスタの状態は変化しない．

【例】

IX＝2345Hであるときに，この命令を実行すると，SP＝2345Hとなる．

LD SP, IY

(Load SP with IY) M=2 T=10(4,6)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	1	1	1	0	0	1	F9H

【命令の機能】 (SP←IY)

インデックス・レジスタIYの内容をスタック・ポインタSPにロードする．

フラグ・レジスタの状態は変化しない．

【例】

IY＝2345Hであるときに，この命令を実行すると，SP＝2345Hとなる．

LD A, I

(Load Acc. with I) M=2 T=9(4, 5)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	0	1	0	1	1	1	57H

【命令の機能】 (A←I)

インタラプト・レジスタIの内容をアキュムレータにロードする．

フラグ・レジスタのP/Vフラグには，IFF2の状態がセットされる．

【例】

割り込み可の状態でI＝00Hであるとき，この命令を実行すると，

Reg.A＝00H，

C	Z	P/V	S	N	H
●	1	1	0	0	0

LD A, R

(Load Acc. with R) M=2 T=9(4,5)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	0	1	1	1	1	1	5FH

【命令の機能】 (A←R)

リフレッシュ・レジスタRの内容を，アキュムレータにロードする．

フラグ・レジスタのP/Vフラグには，IFF2の状態がセットされる．

【例】

R＝58Hで割り込み可状態であるときに，この命令を実行すると，

Reg.A＝58H，

C	Z	P/V	S	N	H
●	0	1	0	0	0

LD I, A

(Load I with Acc.)

M = 2
T = 9 (4, 5)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	0	0	0	1	1	1	47H

【命令の機能】 (I←A)

アキュムレータの内容をインタラプト・レジスタIにロードする．

フラグ・レジスタの状態は変化しない．

【例】

Reg.A＝55Hであるときに，この命令を実行すると，I＝55Hとなる．

LD R, A

(Load R with Acc.)

M = 2
T = 9 (4, 5)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	0	0	1	1	1	1	4FH

【命令の機能】 (R←A)

アキュムレータの内容をリフレッシュ・レジスタRにロードする．

フラグ・レジスタの状態は変化しない．

【例】

Reg.A＝55Hであるときに，この命令を実行すると，R＝55Hとなる．

LDD

(Load location (DE) with location (HL), decrement DE, HL and BC)

M = 4
T = 16 (4, 4, 3, 5)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	0	1	0	0	0	A8H

【命令の機能】 ((DE)←(HL), DE←DE−1, HL←HL−1, BC←BC−1)

レジスタ・ペアHLで示されるアドレスのデータを，レジスタ・ペアDEで示されるアドレスにロードする．その後，レジスタ・ペアDE，HL，BCの内容から1を減じる．

レジスタ・ペアBCの値がゼロになったときには，フラグ・レジスタ中のP/Vフラグがゼロになる．

【例】

① DE＝FF50H，HL＝2345H，BC＝1000H，(2345H)＝55Hであるときに，この命令を実行すると，

(FF50H)＝55H，DE＝FF4FH，HL＝2344H，BC＝0FFFH，

C	Z	P/V	S	N	H
●	●	1	0	0	0

② DE＝FF50H，HL＝2345H，BC＝0001H，(2345H)＝55Hであるときに，この命令を実行すると，

(FF50H)＝55H，DE＝FF4FH，HL＝2344H，BC＝0000H，

C	Z	P/V	S	N	H
●	●	0	●	0	0

LDDR

BC≠0 (M=5, T=21(4,4,3,5,5))　BC=0 (M=4, T=16(4,4,3,5))

(Load location (DE) with location (HL), decrement DE, HL and BC, repeat until BC=0)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	1	1	0	0	0	B8H

【命令の機能】　((DE)←(HL), DE←DE−1, HL←HL−1, BC←BC−1)

レジスタ・ペアHLで示されるアドレスのデータを，レジスタ・ペアDEで示されるアドレスにロードする．その後，レジスタ・ペアDE，HL，BCの内容を1減じる．

以上の過程はBC＝0となるまで繰り返される．

【例】

DE＝3400H，HL＝0723H，BC＝0388Hであり，メモリのデータが次のようになっているとき，この命令を実行すると，

0723H	0722H	0721H		039FH	039EH	039DH	039CH
00H	01H	02H	∫∫	F0H	F1H	F2H	F3H

DE＝3078H，
HL＝039BH，
BC＝0000H，

C	Z	P/V	S	N	H
●	●	0	●	0	0

3400H	33FFH	33FEH		3074H	3073H	3072H	3071H
00H	01H	02H	∫∫	F0H	F1H	F2H	F3H

LDI

(M=4, T=16(4,4,3,5))

(Load location (DE) with(HL),increment DE and HL, decrement BC, repeat until BC=0)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	0	0	0	0	0	A0H

【命令の機能】　((DE)←(HL), DE←DE+1, HL←HL+1, BC←BC−1)

レジスタ・ペアHLで示されるアドレスのデータを，レジスタ・ペアDEで示されるアドレスにロードする．その後レジスタ・ペアDE，およびHLの内容に1を加え，レジスタ・ペアBCから1を減じる．

BC＝0となると，フラグ・レジスタ中のP/Vフラグがゼロになる．

【例】

① DE＝2345H，HL＝0123H，BC＝0321Hであり，(0123H)＝55Hであるときに，この命令を実行すると，(2345H)＝55H，DE＝2346H，HL＝0124H，BC＝0320H，

C	Z	P/V	S	N	H
●	●	1	●	0	0

② DE＝2345H，HL＝0123H，BC＝0001Hであり，(0123H)＝AAHであるときに，この命令を実行すると，(2345H)＝AAH，DE＝2346H，HL＝0124H，BC＝0000H，

C	Z	P/V	S	N	H
●	●	0	●	0	0

LDIR

BC≠0 (M=5, T=21(4,4,3,5,5))　BC=0 (M=4, T=16(4,4,3,5))

(Load location (DE) with (HL), increment DE and HL, decrement BC, repeat until BC=0)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	1	0	0	0	0	B0H

【命令の機能】　((DE)←(HL), DE←DE+1, HL←HL+1, BC←BC−1)

レジスタ・ペアHLで示されるアドレスのデータを，レジスタ・ペアDEで示されるアドレスにロードする．その後レジスタ・ペアDE，およびHLに1を加え，レジスタ・ペアBCから1を減じる．

以上の過程はレジスタ・ペアBCがゼロになるまで繰り返される．

【例】

DE=2345H，HL=0789H，BC=0321Hで，メモリの内容が次のようになっているとき，この命令を実行すると，

0789H	078AH	078BH			0AA7H	0AA8H	0AA9H
01H	02H	03H		～	80H	81H	82H

DE=2666H，
HL=0AAAH，
BC=0000H，

C	Z	P/V	S	N	H
●	●	0	●	0	0

2345H	2346H	2347H			2663H	2664H	2665H
01H	02H	03H		～	80H	81H	82H

NEG

(Negate Acc.)　M=2　T=8(4,4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	0	0	0	1	0	0	44H

【命令の機能】　(A←0−A)

アキュムレータの内容の符号を反転する（ゼロからアキュムレータの内容を引く）．

【例】

① Reg.A=55H(85)であるときに，この命令を実行すると，

Reg.A=ABH(−85)，

C	Z	P/V	S	N	H
1	0	0	1	1	1

② Reg.A=ABH(−85)であるときに，この命令を実行すると，

Reg.A=55H(85)，

C	Z	P/V	S	N	H
1	0	0	0	1	1

NOP

(No operation)　M=1　T=4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	00H

【命令の機能】

何も行われない．フラグ・レジスタの状態も変化しない．

OR (HL)

(Logical OR of location (HL) and Acc.) M=2 T=7(4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	0	1	1	0	1	1	0	B6H

【命令の機能】 (A←A ∨ (HL))

アキュムレータの内容と，レジスタ・ペアHLで示されるアドレスのデータとの論理和をとり，結果をアキュムレータにストアする．

【例】

Reg.A=11H，HL=2345H，(2345H)=23Hであるときに，この命令を実行すると，

Reg.A=33H，

C	Z	P/V	S	N	H
0	0	1	0	0	0

OR (IX+d)

(Logical OR of location (IX+d) and Acc.) M=5 T=19(4,4,3,5,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	0	1	1	0	1	1	0	B6H
←			d				→	

【命令の機能】 (A←A ∨ (IX+d))

インデックス・レジスタIXとディスプレースメントdの和で示されるアドレスのデータと，アキュムレータの内容との論理和をとり，結果をアキュムレータにストアする．

【例】

Reg.A=11H，IX=2345H，d=67H，(23ACH)=23Hであるときに，この命令を実行すると，

Reg.A=33H，

C	Z	P/V	S	N	H
0	0	1	0	0	0

OR (IY+d)

(Logical OR of location (IY+d) and Acc.) M=5 T=19(4,4,3,5,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	0	1	1	0	1	1	0	B6H
←			d				→	

【命令の機能】 (A←A ∨ (IY+d))

インデックス・レジスタIYとディスプレースメントdの和で示されるアドレスのデータと，アキュムレータの内容との論理和をとり，結果をアキュムレータにストアする．

【例】

Reg.A=11H，IY=2345H，d=67H，(23ACH)=23Hであるときに，この命令を実行すると，

Reg.A=33H，

C	Z	P/V	S	N	H
0	0	1	0	0	0

OR r

(Logical OR of Reg. r and Acc.)　M = 1　T = 4

【オブジェクト・コード】

7	6	5	4	3	2	1	0
1	0	1	1	0	← r →		

r	r の値	ニーモニック	オブジェクト・コード
B	000	OR B	B0H
C	001	OR C	B1H
D	010	OR D	B2H
E	011	OR E	B3H
H	100	OR H	B4H
L	101	OR L	B5H
A	111	OR A	B7H

【命令の機能】　(A←A ∨ r)

レジスタ r とアキュムレータの内容との論理和をとり，結果をアキュムレータにストアする．

【例】

Reg.A＝11H，Reg.D＝23H であるときに，OR D 命令を実行すると，

Reg.A＝33H，

C	Z	P/V	S	N	H
0	0	1	0	0	0

OR n

(Logical OR of n and Acc.)　M = 2　T = 7(4,3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	0	1	1	0	F6H
← n →								

【命令の機能】　(A←A ∨ n)

アキュムレータの内容とイミーディエイト・データ n との論理和をとり，結果をアキュムレータにストアする．

【例】

Reg.A＝11H のとき，OR 23H 命令を実行すると，

Reg.A＝33H，

C	Z	P/V	S	N	H
0	0	1	0	0	0

OTDR

B≠0 (M = 5, T = 21(4,5,3,4,5))　B = 0 (M = 4, T = 16(4,5,3,4))

(Load output port (C) with location (HL), decrement HL and B, repeat until B＝0)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	1	1	0	1	1	BBH

【命令の機能】

((C)←(HL)，B←B－1，HL←HL－1)

レジスタ・ペア HL で示されるアドレスのデータを，レジスタ C で示される出力ポートにロードする．その後，レジスタ B およびレジスタ・ペア HL の内容を 1 減じる．

以上の過程をレジスタ B がゼロになるまで繰り返す．

【例】

Reg.B＝10H，Reg.C＝55H，HL＝2345H で，メモリの内容が次のようになっているとき，この命令を実行すると，

2345H	2344H	2343H	…	2339H	2338H	2337H	2336H
00H	01H	02H	…	0CH	0DH	0EH	0FH

Reg.B＝00H，
Reg.HL＝2335H，

C	Z	P/V	S	N	H
●	1	X	X	1	●

出力ポート 55H には 00H ～0FH の 16 バイトのデータが出力される．

OTIR

B≠0 (M=5, T=21(4, 5, 3, 4, 5))　B=0 (M=4, T=16(4, 5, 3, 4))

(Load output port (C) with location (HL), decrement B and increment HL, repeat until B=0)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	1	0	0	1	1	B3H

【命令の機能】 ((C)←(HL), B←B−1, HL←HL+1)

レジスタ・ペアHLで示されるアドレスのデータを，レジスタCで示される出力ポートにロードする．その後，レジスタBの内容を1減じ，レジスタ・ペアHLの内容に1を加える．以上の過程は，レジスタBの値がゼロになるまで繰り返される．

【例】

Reg.B=10H，Reg.C=55H，HL=2345Hで，メモリの内容が次のようになっているときに，この命令を実行すると，

2345H	2346H	2347H		2351H	2352H	2353H	2354H
00H	01H	02H	～	0CH	0DH	0EH	0FH

Reg.B=00H，
Reg.HL=2355H，

C	Z	P/V	S	N	H
●	1	X	X	1	●

出力ポート55Hには，00H～0FHの16バイトのデータが出力される．

OUT (C), r

(Load output port (C) with r)　M=3　T=12(4, 4, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	←	r	→	0	0	1	

r	rの値	ニーモニック	オブジェクト・コード
B	0 0 0	OUT (c), B	4 1 H
C	0 0 1	OUT (c), C	4 9 H
D	0 1 0	OUT (c), D	5 1 H
E	0 1 1	OUT (c), E	5 9 H
H	1 0 0	OUT (c), H	6 1 H
L	1 0 1	OUT (c), L	6 9 H
A	1 1 1	OUT (c), A	7 9 H

【命令の機能】 ((C)←r)

レジスタCで示される出力ポートに，レジスタrの内容をロードする．

フラグ・レジスタの状態は変化しない．

【例】

Reg.C=55H，Reg.E=AAHであるときに，OUT(C)，E命令を実行すると，
出力ポート55HにデータAAHがロードされる．

OUT (n), A

(Load output port (n) with Acc.)　M=3　T=11(4, 3, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	0	0	1	1	D3H
←			n				→	

【命令の機能】 ((n)←A)

アキュムレータの内容を，出力ポートnにロードする．

フラグ・レジスタの状態は変化しない．

【例】

Reg.A=55Hであるときに，OUT (AAH)，A命令を実行すると，出力ポートAAHにデータ55Hがロードされる．

OUTD

(Load output port (C) with location (HL), decrement B and HL)

M＝4
T＝16(4, 5, 3, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	0	1	0	1	1	ABH

【命令の機能】 ((C)←(HL), B←B－1, HL←HL－1)

レジスタ・ペア HL で示されるアドレスのデータを，レジスタ C で示される出力ポートにロードする．その後レジスタ B，およびレジスタ・ペア HL の内容が 1 減じられる．

【例】

① Reg.B＝10H, Reg.C＝55H, HL＝2345H, (2345H)＝AAH であるときに，この命令を実行すると，出力ポート 55H にデータ AAH がロードされ，

Reg.B＝0FH,
Reg.HL＝2344H,

C	Z	P/V	S	N	H
●	0	X	X	1	X

② Reg.B＝01H, Reg.C＝55H, HL＝2345H, (2345H)＝AAH であるときに，この命令を実行すると，出力ポート 55H にデータ AAH がロードされ，

Reg.B＝00H,
Reg.HL＝2344H,

C	Z	P/V	S	N	H
●	1	X	X	1	X

OUTI

(Load output port (C) with location (HL), decrement B and increment HL)

M＝4
T＝16(4, 5, 3, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
1	0	1	0	0	0	1	1	A3H

【命令の機能】 ((C)←(HL), B←B－1, HL←HL＋1)

レジスタ・ペア HL で示されるアドレスのデータを，レジスタ C で示される出力ポートにロードする．その後，レジスタ B の内容を 1 減じ，レジスタ・ペア HL の内容に 1 を加える．

【例】

① Reg.B＝10H, Reg.C＝55H, HL＝2345H, (2345H)＝AAH であるときに，この命令を実行すると，出力ポート 55H にデータ AAH がロードされ，

Reg.B＝0FH,
Reg.HL＝2346H,

C	Z	P/V	S	N	H
●	0	X	X	1	X

② Reg.B＝01H, Reg.C＝55H, HL＝2345H, (2345H)＝AAH であるときに，この命令を実行すると，出力ポート 55H にデータ AAH がロードされ，

Reg.B＝00H,
Reg.HL＝2346H,

C	Z	P/V	S	N	H
●	1	X	X	1	X

POP ss

(Load ss with top of stack)

M＝3
T＝10(4, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0
1	1	s	s	0	0	0	1

ss	ss の値	ニーモニック	オブジェクト・コード
BC	0 0	POP BC	C 1 H
DE	0 1	POP DE	D 1 H
HL	1 0	POP HL	E 1 H
AF	1 1	POP AF	F 1 H

【命令の機能】

$(SS_L \leftarrow (SP),\ SS_H \leftarrow (SP+1),\ SP \leftarrow SP+2)$

スタックの内容をレジスタ・ペアにロードする．

レジスタ・ペアの上位バイトには SP＋1 のデータがロードされ，下位バイトには SP が示すデータがロードされる．スタック・ポインタ SP には 2 が加えられる．

フラグ・レジスタの状態は POP AF 命令を実行したときのみ，変化する．

【例】

① SP＝2345H, (2345H)＝67H, (2346H)＝89H であるとき，POP BC 命令を実行すると，

Reg.B＝89H, Reg.C＝67H, SP＝2347H

② SP＝1000H, (1000H)＝ABH, (1001H)＝CDH, であるとき，POP AF 命令を実行すると，

Reg.A＝CDH, Reg.F＝ABH, SP＝1002H

POP IX

(Load IX with top of stack)

M=4
T=14(4, 4, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	1	0	0	0	0	1	E1H

【命令の機能】

$(IX_L \leftarrow (SP),\ IX_H \leftarrow (SP+1),\ SP \leftarrow SP+2)$

スタックの内容をインデックス・レジスタ IX にロードする。

IX の上位バイトには SP+1 のデータがロードされ、下位バイトには SP が示すデータがロードされる。スタック・ポインタ SP には 2 が加えられる。

フラグ・レジスタの状態は変化しない。

【例】

SP=2345H, (2345H)=67H, (2346H)=89H であるときに、この命令を実行すると、
IX=8967H (IX_H=89H, IX_L=67H), SP=2347H となる。

POP IY

(Load IY with top of stack)

M=4
T=14(4, 4, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	1	0	0	0	0	1	E1H

【命令の機能】

$(IY_L \leftarrow (SP),\ IY_H \leftarrow (SP+1),\ SP \leftarrow SP+2)$

スタックの内容をインデックス・レジスタ IY にロードする。

IY の下位バイトには SP が示すアドレスのデータがロードされ、上位バイトには SP+1 が示すアドレスのデータがロードされる。スタック・ポインタ SP には 2 が加えられる。

フラグ・レジスタの状態は変化しない。

【例】

SP=2345H, (2345H)=67H, (2346H)=89H であるときに、この命令を実行すると、
IY=8967H (IY_H=89H, IY_L=67H), SP=2347H となる。

PUSH ss

(Load Reg. pair ss onto stack)

M=3
T=11(5, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0
1	1	s	s	0	1	0	1

ss	ssの値	ニーモニック	オブジェクト・コード
BC	0 0	PUSH BC	C 5 H
DE	0 1	PUSH DE	D 5 H
HL	1 0	PUSH HL	E 5 H
AF	1 1	PUSH AF	F 5 H

【命令の機能】

$((SP-1) \leftarrow ss_H,\ (SP-2) \leftarrow ss_L,\ SP \leftarrow SP-2)$

レジスタ・ペア ss の内容をスタック上にロードする。

ss の上位バイトは SP−1 が示すアドレスにロードされ、下位バイトは SP−2 が示すアドレスにロードされる。スタック・ポインタ SP は 2 だけ減じられる。

フラグ・レジスタの状態は変化しない。

【例】

① SP=2345H, Reg.B=67H, Reg.C=89H であるときに、PUSH BC 命令を実行すると、
(2344H)=67H, (2343H)=89H, SP=2343H となる。

② SP=1000H, Reg.A=67H, Reg.F=89H であるときに、PUSH AF 命令を実行すると、
(0FFFH)=67H, (0FFEH)=89H, SP=0FFEH となる。

PUSH IX

(Load IX onto stack)　M=3　T=15(4, 5, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	1	0	0	1	0	1	E5H

【命令の機能】

$((SP-1) \leftarrow IX_H,\ (SP-2) \leftarrow IX_L,\ SP \leftarrow SP-2)$

インデックス・レジスタIXの内容を，スタックにロードする．

IXの上位バイトはSP−1が示すアドレスにロードされ，下位バイトはSP−2が示すアドレスにロードされる．スタック・ポインタSPは2だけ減じられる．

フラグ・レジスタの状態は変化しない．

【例】

IX=6789H, SP=2345Hであるときに，この命令を実行すると，

(2344H)=67H, (2343H)=89H, SP=2343Hとなる．

PUSH IY

(Load IY onto stack)　M=4　T=15(4, 5, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	1	0	0	1	0	1	E5H

【命令の機能】

$((SP-1) \leftarrow IY_H,\ (SP-2) \leftarrow IY_L,\ SP \leftarrow SP-2)$

インデックス・レジスタIYの内容をスタックにロードする．

IYの上位バイトはSP−1が示すアドレスにロードされ，下位バイトはSP−2が示すアドレスにロードされる．スタック・ポインタSPは2だけ減じられる．

フラグ・レジスタの状態は変化しない．

【例】

IY=6789H, SP=2345Hであるときに，この命令を実行すると，

(2344H)=67H, (2343H)=89H, SP=2343Hとなる．

RES b, (HL)

(Reset bit b of location (HL))　M=4　T=15(4, 4, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
1	0	←b	→		1	1	0	

ビット	bの値	ニーモニック	オブジェクト・コード
0	0 0 0	RES 0, (HL)	8 6 H
1	0 0 1	RES 1, (HL)	8 E H
2	0 1 0	RES 2, (HL)	9 6 H
3	0 1 1	RES 3, (HL)	9 E H
4	1 0 0	RES 4, (HL)	A 6 H
5	1 0 1	RES 5, (HL)	A E H
6	1 1 0	RES 6, (HL)	B 6 H
7	1 1 1	RES 7, (HL)	B E H

【命令の機能】　$((HL)_b \leftarrow 0)$

レジスタ・ペアHLで示されるアドレスのデータのビットb(0〜7)をゼロにする．

フラグ・レジスタの状態は変化しない．

【例】

HL=2345H, (2345H)=55Hであるときに，RES 2, (HL)命令を実行すると，

(2345H)=51Hとなる．

7	6	5	4	3	2	1	0
0	1	0	1	0	1	0	1
					↑ 0		

RES b, (IX+d)

(Reset bit b of location (IX+d))　M=6　T=23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	0	0	1	0	1	1	CBH
←—	—	—	d	—	—	—	—→	
1	0	←—	b	—→	1	1	0	

ビット	bの値	ニーモニック	オブジェクト・コード
0	0 0 0	RES 0, (IX+d)	8 6 H
1	0 0 1	RES 1, (IX+d)	8 E H
2	0 1 0	RES 2, (IX+d)	9 6 H
3	0 1 1	RES 3, (IX+d)	9 E H
4	1 0 0	RES 4, (IX+d)	A 6 H
5	1 0 1	RES 5, (IX+d)	A E H
6	1 1 0	RES 6, (IX+d)	B 6 H
7	1 1 1	RES 7, (IX+d)	B E H

【命令の機能】　$((IX+d)_b \leftarrow 0)$

インデックス・レジスタIXとディスプレースメントdの和で示されるアドレスのデータのビットbをゼロにする．

フラグ・レジスタの状態は変化しない．

【例】

IX＝2345H，(23ACH)＝55Hであるときに，RES 6, (IX+67H)命令を実行すると，
(23ACH)＝15Hとなる．

RES b, (IY+d)

(Reset bit b of location (IY+d))　M=6　T=23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	0	0	1	0	1	1	CBH
←—	—	—	d	—	—	—	—→	
1	0	←—	b	—→	1	1	0	

ビット	bの値	ニーモニック	オブジェクト・コード
0	0 0 0	RES 0, (IY+d)	8 6 H
1	0 0 1	RES 1, (IY+d)	8 E H
2	0 1 0	RES 2, (IY+d)	9 6 H
3	0 1 1	RES 3, (IY+d)	9 E H
4	1 0 0	RES 4, (IY+d)	A 6 H
5	1 0 1	RES 5, (IY+d)	A E H
6	1 1 0	RES 6, (IY+d)	B 6 H
7	1 1 1	RES 7, (IY+d)	B E H

【命令の機能】　$((IY+d)_b \leftarrow 0)$

インデックス・レジスタIYとディスプレースメントdの和で示されるアドレスのデータのビットbを0にする．

フラグ・レジスタの状態は変化しない．

【例】

IY＝2345H，(23ACH)＝55Hであるときに，RES 0, (IY+67H)命令を実行すると，
(23ACH)＝54Hとなる．

RES b, r

(Reset bit b of Reg.r)　M=2　T=8(4, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
1	0	←b→			←r→			

		レジスタr						
		A	B	C	D	E	H	L
ビットb	0	87	80	81	82	83	84	85
	1	8F	88	89	8A	8B	8C	8D
	2	97	90	91	92	93	94	95
	3	9F	98	99	9A	9B	9C	9D
	4	A7	A0	A1	A2	A3	A4	A5
	5	AF	A8	A9	AA	AB	AC	AD
	6	B7	B0	B1	B2	B3	B4	B5
	7	BF	B8	B9	BA	BB	BC	BD

【命令の機能】　$(r_b \leftarrow 0)$

レジスタrのビットbをゼロにする．
フラグ・レジスタの状態は変化しない．

【例】

① Reg.B=55Hであるときに，RES　2，B命令を実行すると，
Reg.B=51Hとなる．

② Reg.L=AAHであるときに，RES　5，L命令を実行すると，
Reg.L=8AHとなる．

RET

(Return from subroutine)　M=3　T=10(4, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	0	1	C9H

【命令の機能】

$(PC_L \leftarrow (SP),\ PC_H \leftarrow (SP+1),\ SP \leftarrow SP+2)$

スタック・ポインタSPが示すアドレスのデータを，プログラム・カウンタPCの下位バイトにロードし，SP+1が示すアドレスのデータをPCの上位バイトにロードする．SPには2が加えられる．フラグ・レジスタの状態は変化しない．

この命令は，スタック上にリターン・アドレスがプッシュされているときに，サブルーチンからコール先へ戻るときに使われる．

【例】

SP=2000Hとし，1000H番地でCALL　XXX命令を実行すると，スタック上にはCALL命令の次のアドレス(1003H)がプッシュされ，(1FFFH)=10H，(1FFEH)=03H，SP=1FFEHとなり，サブルーチンXXXへジャンプする．

サブルーチンXXXの終りでRET命令を実行すると，SPが示すアドレス(1FFEH)のデータ(03H)をPCの下位バイトにロードし，SP+1が示すアドレス(1FFFH)のデータ(10H)をPCの上位バイトにロードする．

したがってPC=1003Hとなり，CALL命令の次のアドレスへ戻れることになる．SPの値は2000Hになる．

RET cc

(Return from subroutine if condition is true)　M=3　T=11(5, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0
1	1	←	cc	→	0	0	0

cc	ccの値	ニーモニック	オブジェクト・コード
NZ	0 0 0	RET NZ	C 0 H
Z	0 0 1	RET Z	C 8 H
NC	0 1 0	RET NC	D 0 H
C	0 1 1	RET C	D 8 H
PO	1 0 0	RET PO	E 0 H
PE	1 0 1	RET PE	E 8 H
P	1 1 0	RET P	F 0 H
M	1 1 1	RET M	F 8 H

【命令の機能】

$(PC_L \leftarrow (SP),\ PC_H \leftarrow (SP+1),\ SP \leftarrow SP+2)$

判定条件が満足した場合には，スタック・ポインタSPが示すアドレスの内容を，プログラム・カウンタPCの下位バイト(PC_L)にロードし，SP+1が示すアドレスの内容をPCの上位バイト(PC_H)にロードする．SPには2が加えられる．

判定条件が満足しない場合には，次の命令を実行する．フラグ・レジスタの状態は変化しない．

【例】

(SP)=45H，(SP+1)=23HでZフラグが"1"であるときに，RET　Z命令を実行すると，2345H番地にジャンプする．

RETI

(Return from interrupt)　M=4　T=14(4, 4, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	0	0	1	1	0	1	4DH

【命令の機能】

$(PC_L \leftarrow (SP),\ PC_H \leftarrow (SP+1),\ SP \leftarrow SP+2)$

スタック・ポインタSPが示すアドレスの内容を，プログラム・カウンタPCの下位バイト(PC_L)にロードし，SP+1が示すアドレスの内容をPCの上位バイト(PC_H)にロードする．SPには2が加えられる．フラグ・レジスタの状態は変化しない．

この命令は割り込みルーチンの最後に用いられ，割り込まれたプログラムに戻るときに使われる．また，この命令は割り込みディジー・チェインの制御を行う．

【例】

(SP)=00H，(SP+1)=20Hであるときに，この命令を実行すると，2000H番地にジャンプする．

RETN

(Return from non-maskable interrupt)　M=4　T=14(4, 4, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	0	0	0	1	0	1	45H

【命令の機能】　$(PC_L \leftarrow (SP),\ PC_H \leftarrow (SP+1),$
$IFF1 \leftarrow IFF2,\ SP \leftarrow SP+2)$

スタック・ポインタSPが示すアドレスの内容を，プログラム・カウンタPCの下位バイト(PC_L)にロードし，SP+1が示すアドレスの内容をPCの上位バイト(PC_H)にロードし，IFF2の状態をIFF1にセットする．SPには2が加えられる．フラグ・レジスタの状態は変化しない．

この命令はNMIルーチンの最後に実行され，NMIにより割り込まれたプログラムに戻るとともに，割り込みイネーブル・フリップフロップの状態をNMIが発生する前の状態に戻す．

【例】

(SP)=00H，(SP+1)=20H，IFF1=0，IFF2=1であるときに，この命令を実行すると，2000H番地にジャンプし，IFF1=IFF2=1となる．

RL (HL)

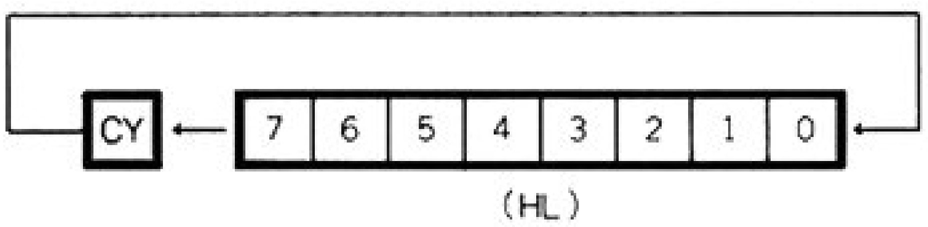
(Rotate left location (HL) through carry)

M=4
T=15(4, 4, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	0	0	1	0	1	1	0	16H

【命令の機能】



レジスタ・ペアHLで示されるアドレスのデータを，1ビット左へシフトする．

データのビット7はキャリー・フラグに入り，キャリー・フラグの状態はデータのビット0に入る．

【例】

HL=2345H，(2345H)=55H，CY=1であるときに，この命令を実行すると，

(2345H)=ABH，

C	Z	P/V	S	N	H
0	0	0	1	0	0

CY		7	6	5	4	3	2	1	0
1		0	1	0	1	0	1	0	1

⬇

CY		7	6	5	4	3	2	1	0
0		1	0	1	0	1	0	1	1

RL (IX+d)

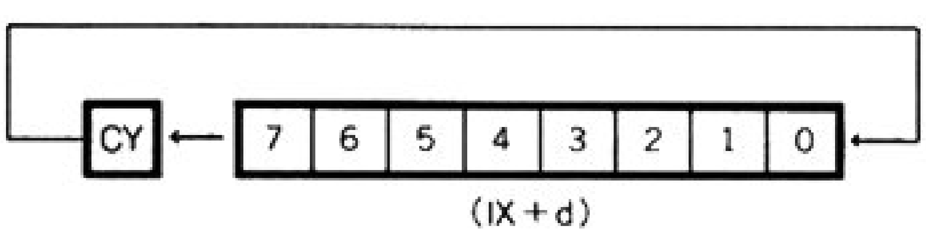
(Rotate left location (IX+d) through carry)

M=6
T=23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	0	0	1	0	1	1	CBH
←			d				→	
0	0	0	1	0	1	1	0	16H

【命令の機能】



インデックス・レジスタIXと，ディスプレースメントdの和で示されるアドレスのデータを，1ビット左へシフトする．

データのビット7はキャリー・フラグに入り，キャリー・フラグの状態はデータのビット0に入る．

【例】

IX=2345H，d=67H，(23ACH)=AAH，CY=0のときに，この命令を実行すると，

(23ACH)=54H，

C	Z	P/V	S	N	H
1	0	0	0	0	0

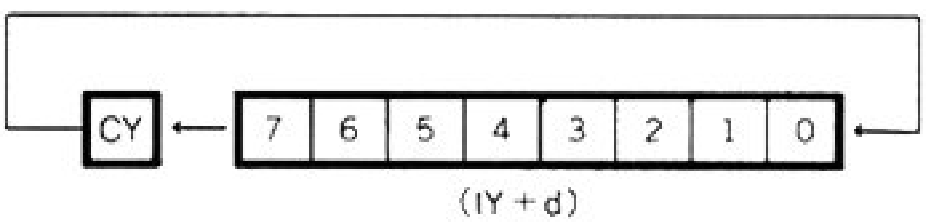
RL (IY+d)

(Rotate left location (IY+d) through carry)　M=6　T=23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	0	0	1	0	1	1	CBH
←			d				→	
0	0	0	1	0	1	1	0	16H

【命令の機能】



インデックス・レジスタIYとディスプレースメントdの和で示されるアドレスのデータを1ビット左にシフトする.

データのビット7はキャリー・フラグに入り，キャリー・フラグの状態はデータのビット0になる.

【例】

IY=2345H，d=67H，(23ACH)=AAH，CY=1であるときに，この命令を実行すると，

(23ACH)=55H，

C	Z	P/V	S	N	H
1	0	1	0	0	0

RL r

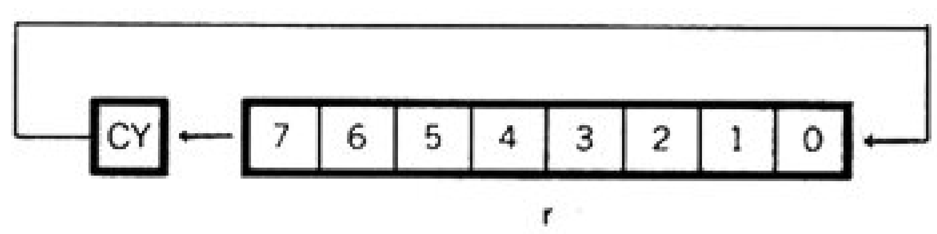
(Rotate left Reg.r through carry)　M=2　T=8(4, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	0	0	1	0	←	r	→	

r	rの値	ニーモニック	オブジェクト・コード
B	000	RL B	10H
C	001	RL C	11H
D	010	RL D	12H
E	011	RL E	13H
H	100	RL H	14H
L	101	RL L	15H
A	111	RL A	17H

【命令の機能】



レジスタrの内容を1ビット左にシフトする．レジスタrのビット7はキャリー・フラグに入り，キャリー・フラグの状態はレジスタrのビット0に入る.

【例】

Reg.E=FFH，CY=0のとき，RL　E命令を実行すると，

Reg.E=FEH，

C	Z	P/V	S	N	H
1	0	0	1	0	0

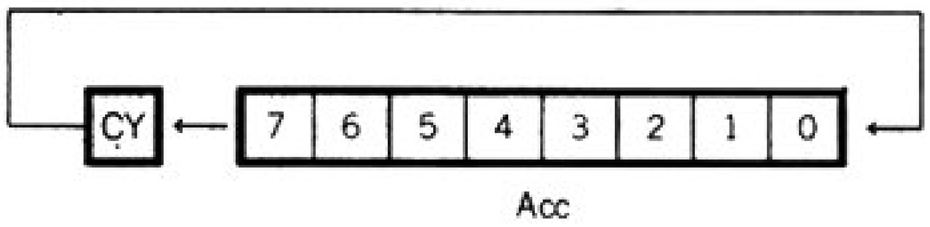
RLA

(Rotate left Acc. through carry)　M＝1　T＝4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	0	1	0	1	1	1	17H

【命令の機能】



アキュムレータの内容を1ビット左にシフトする。アキュムレータのビット7はキャリー・フラグに入り，キャリー・フラグはアキュムレータのビット0に入る。

この命令の機能はRL　r命令のr＝Aの場合と同じだが，この命令のほうが命令の長さも短く，実行速度も速い。

ただし，フラグ・レジスタの変化が少し異なる。

【例】

Reg.A＝AAH，CY＝1であるときに，この命令を実行すると，

Reg.A＝55H，

C	Z	P/V	S	N	H
1	●	●	●	0	0

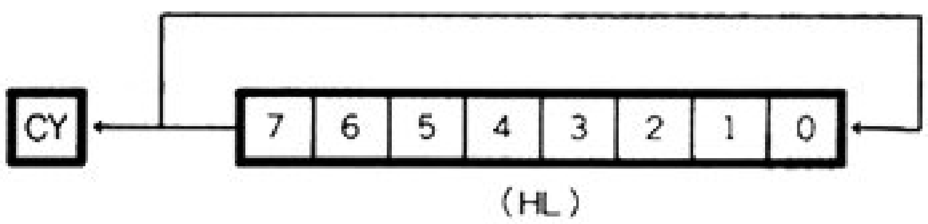
RLC　(HL)

(Rotate left location (HL) circular)　M＝4　T＝15(4, 4, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	0	0	0	0	1	1	0	06H

【命令の機能】



レジスタ・ペアHLで示されるアドレスのデータを1ビット左にシフトする。

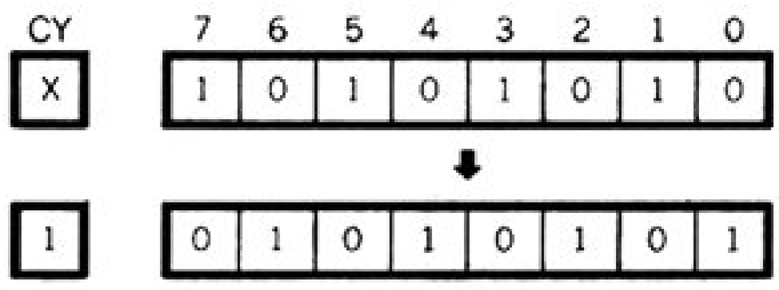
データのビット7はキャリー・フラグ，およびデータのビット0に入る。

【例】

HL＝2345H，(2345H)＝AAHであるときに，この命令を実行すると，

(2345H)＝55H，

C	Z	P/V	S	N	H
1	0	1	0	0	0



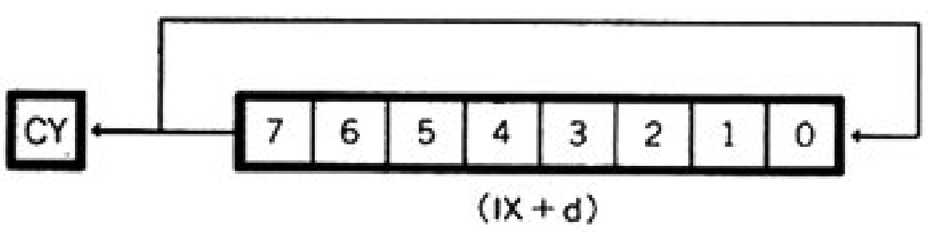
RLC　(IX+d)

(Rotate left location (IX+d) circular)　M＝6　T＝23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	0	0	1	0	1	1	CBH
←d→								
0	0	0	0	0	1	1	0	06H

【命令の機能】



インデックス・レジスタIXとディスプレースメントdの和で示されるアドレスのデータを1ビット左にシフトする。

データのビット7はキャリー・フラグ，およびデータのビット0に入る。

【例】

IX＝2345H，d＝67H，(23ACH)＝AAHであるときに，この命令を実行すると，

(23ACH)＝55H，

C	Z	P/V	S	N	H
1	0	1	0	0	0

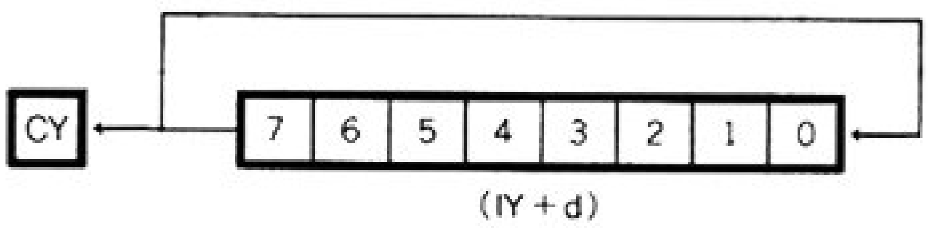
RLC (IY+d)

(Rotate left location (IY+d) circular)　M=6　T=23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	0	0	1	0	1	1	CBH
←			d				→	
0	0	0	0	0	1	1	0	06H

【命令の機能】



インデックス・レジスタ IY とディスプレースメント d の和で示されるアドレスのデータを 1 ビット左にシフトする.

データのビット 7 はキャリー・フラグとデータのビット 0 に入る.

【例】

IY=2345H, d=67H, (23ACH)=55H であるときに, この命令を実行すると,

(23ACH)=AAH,

C	Z	P/V	S	N	H
0	0	1	1	0	0

RLC r

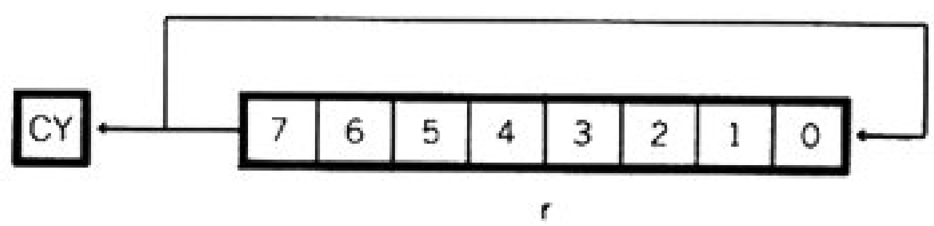
(Rotate Reg.r circular)　M=2　T=8(4, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	0	0	0	0	←	r	→	

r	rの値	ニーモニック	オブジェクト・コード
B	000	RLC B	00H
C	001	RLC C	01H
D	010	RLC D	02H
E	011	RLC E	03H
H	100	RLC H	04H
L	101	RLC L	05H
A	111	RLC A	07H

【命令の機能】



レジスタ r の内容を 1 ビット左にシフトする. r のビット 7 はキャリー・フラグと r のビット 0 に入る.

【例】

Reg.E=AAH であるときに, RLC E 命令を実行すると,

Reg.E=55H,

C	Z	P/V	S	N	H
1	0	1	0	0	0

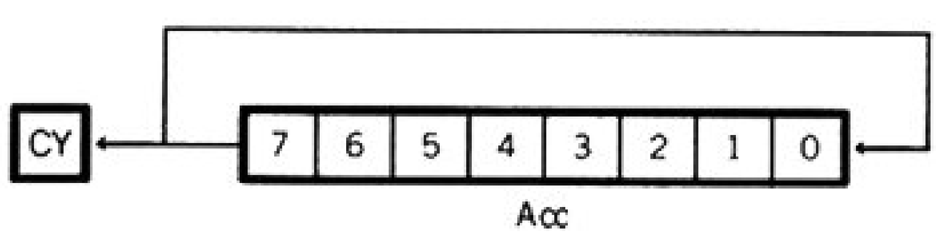
RLCA

(Rotate left Acc. circular)　M=1　T=4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	0	0	0	1	1	1	07H

【命令の機能】



アキュムレータの内容を, 1 ビット左にシフトする. アキュムレータのビット 7 はキャリー・フラグとアキュムレータのビット 0 に入る.

この命令の機能は RLC r 命令の r=A の場合と同じであるが, この命令のほうが命令の長さが短く, 実行速度も速い. ただし, フラグ・レジスタの変化が少し異なる.

【例】

Reg.A=55H であるときに, この命令を実行すると,

Reg.A=AAH,

C	Z	P/V	S	N	H
0	●	●	●	0	0

RLD

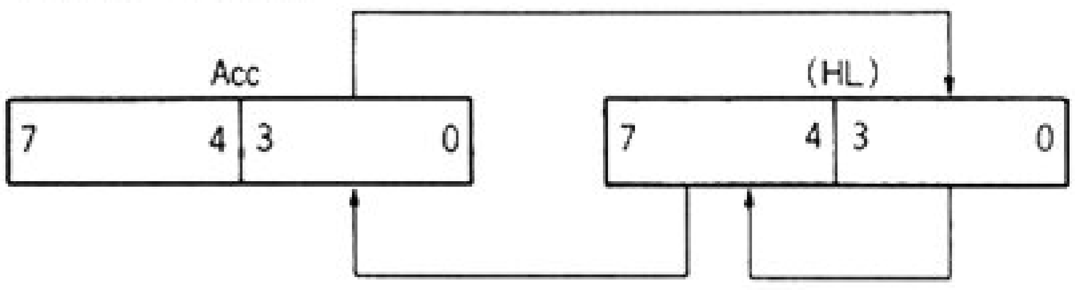
(Rotate digit left and right between Acc. and location (HL))

M＝5
T＝18(4, 4, 3, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	1	0	1	1	1	1	6FH

【命令の機能】



レジスタ・ペアHLで示されるアドレスのデータの下位4ビットを上位4ビットに移動し，上位4ビットをアキュムレータの下位4ビットに移動する．

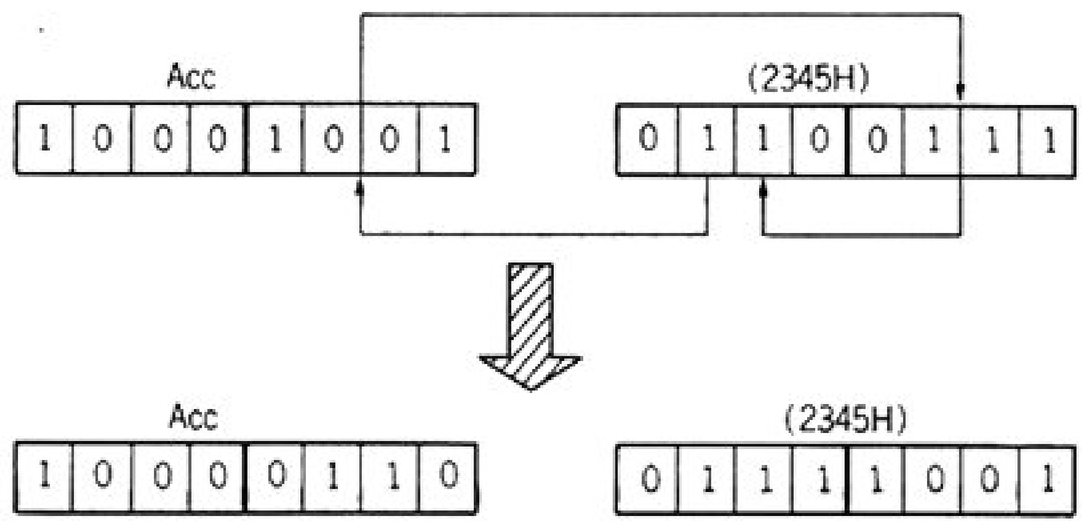
アキュムレータの下位4ビットは，データの下位4ビットに移動される．アキュムレータの上位4ビットの状態は変化しない．

【例】

HL＝2345H，(2345H)＝67H，Reg.A＝89Hであるときに，この命令を実行すると，

(2345H)＝79H，
RegA＝86H，

C	Z	P/V	S	N	H
●	0	0	1	0	0



RR (HL)

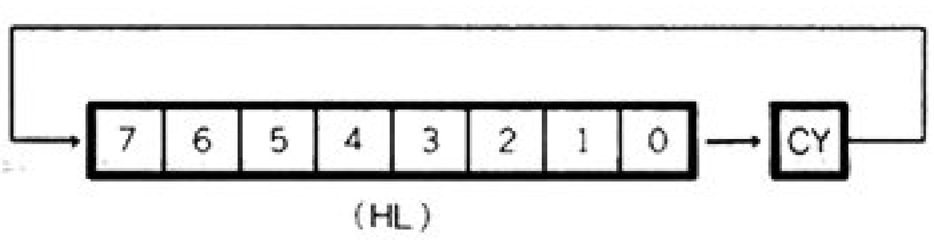
(Rotate right location (HL) through carry)

M＝4
T＝15(4, 4, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	0	0	1	1	1	1	0	1EH

【命令の機能】



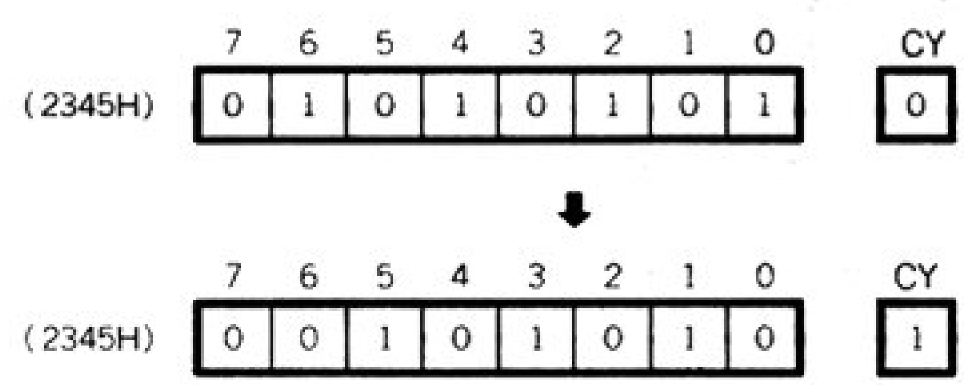
レジスタ・ペアHLで示されるアドレスのデータを，1ビット右にシフトする．データのビット0はキャリー・フラグに入り，キャリー・フラグの状態はデータのビット7に入る．

【例】

HL＝2345H，(2345H)＝55H，C＝0であるときに，この命令を実行すると，

(2345H)＝2AH，

C	Z	P/V	S	N	H
1	0	0	0	0	0



RR (IX+d)

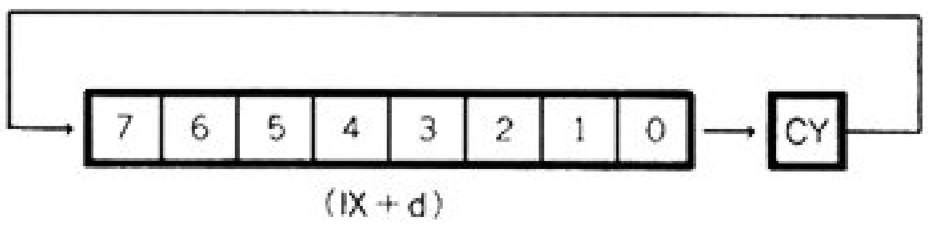
(Rotate right location (IX+d) through carry)

M=6
T=23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	0	0	1	0	1	1	CBH
←d→								
0	0	0	1	1	1	1	0	1EH

【命令の機能】



インデックス・レジスタIXとディスプレースメントdの和で示されるアドレスのデータを，1ビット右にシフトする．

データのビット0はキャリー・フラグに入り，キャリー・フラグの状態はデータのビット7に入る．

【例】

IX=2345H，d=67H，(23ACH)=55H，CY=0であるときに，この命令を実行すると，

(23ACH)=2AH，

C	Z	P/V	S	N	H
1	0	0	0	0	0

RR (IY+d)

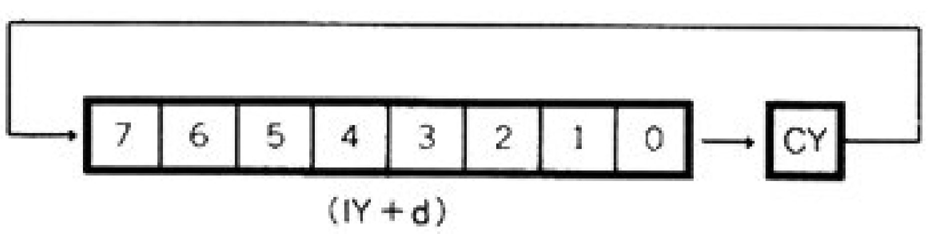
(Rotate right location (IY+d) through carry)

M=6
T=23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	0	0	1	0	1	1	CBH
←d→								
0	0	0	1	1	1	1	0	1EH

【命令の機能】



インデックス・レジスタIYとディスプレースメントdの和で示されるアドレスのデータを，1ビット右にシフトする．

データのビット0はキャリー・フラグに入り，キャリー・フラグの状態はデータのビット7に入る．

【例】

IY=2345H，d=ABH，(22F0H)=55H，CY=0であるときに，この命令を実行すると，

(22F0H)=2AH，

C	Z	P/V	S	N	H
1	0	0	0	0	0

RR r

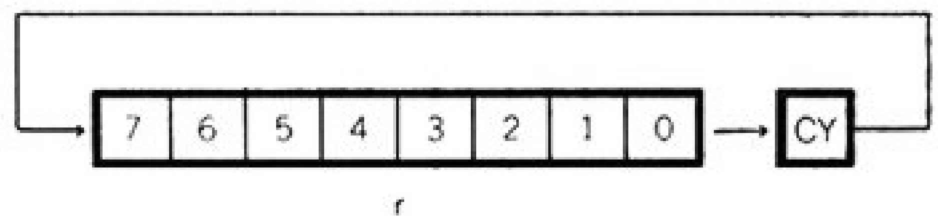
M＝2
(Rotate right Reg. r through carry)　T＝8(4, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	0	0	1	1	←r→			

r	rの値	ニーモニック	オブジェクト・コード
B	000	RR B	18H
C	001	RR C	19H
D	010	RR D	1AH
E	011	RR E	1BH
H	100	RR H	1CH
L	101	RR L	1DH
A	111	RR A	1FH

【命令の機能】



レジスタ r の内容を 1 ビット右にシフトする．レジスタ r のビット 0 は，キャリー・フラグに入り，キャリー・フラグの状態はレジスタ r のビット 7 に入る．

【例】

Reg.E＝55H，CY＝0 であるときに，RR　E 命令を実行すると，

Reg.E＝2AH，

C	Z	P/V	S	N	H
1	0	0	0	0	0

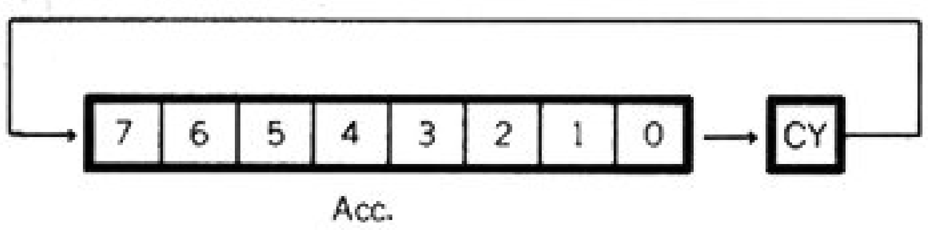
RRA

M＝1
(Rotate right Acc. through carry)　T＝4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	0	1	1	1	1	1	1FH

【命令の機能】



アキュムレータの内容を 1 ビット右にシフトする．アキュムレータのビット 0 はキャリー・フラグに入り，キャリー・フラグの状態はアキュムレータのビット 7 に入る．

この命令の機能は，RR　r 命令の場合の r＝A のときと同じであるが，この命令のほうが命令の長さが短く，実行速度も速い．ただし，フラグ・レジスタの変化は少し異なる．

【例】

Reg.A＝55H，CY＝0 であるときに，この命令を実行すると，

Reg.A＝2AH，

C	Z	P/V	S	N	H
1	●	●	●	0	0

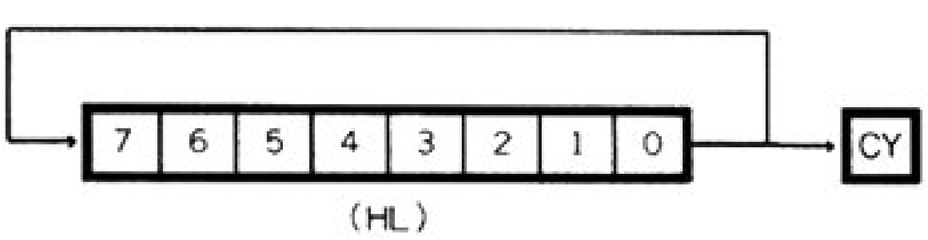
RRC (HL)

M＝4
(Rotate right location (HL) circular)　T＝15(4, 4, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	0	0	0	1	1	1	0	OEH

【命令の機能】



レジスタ・ペア HL で示されるアドレスのデータを，1 ビット右にシフトする．データのビット 0 はデータのビット 7 とキャリー・フラグに入る．

【例】

HL＝2345H，(2345H)＝55H であるときに，この命令を実行すると，

(2345H)＝AAH，

C	Z	P/V	S	N	H
1	0	1	1	0	0

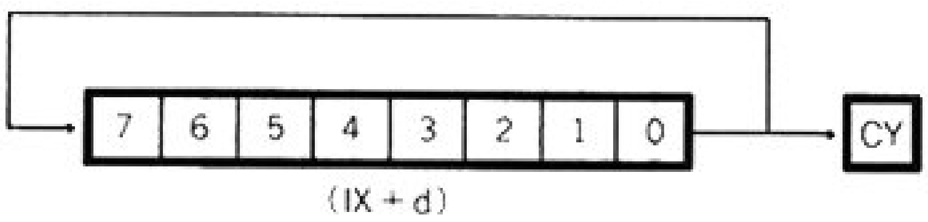
RRC (IX+d)

(Rotate right location (IX+d) circular)　　M＝6　T＝23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	0	0	1	0	1	1	CBH
← d →								
0	0	0	0	1	1	1	0	OEH

【命令の機能】



インデックス・レジスタIXとディスプレースメントdの和で示されるアドレスのデータを1ビット右にシフトする。

データのビット0はキャリー・フラグとデータのビット7に入る。

【例】

IX＝2345H, d＝67H, (23ACH)＝55Hであるときに、この命令を実行すると、

(23ACH)＝AAH,

C	Z	P/V	S	N	H
1	0	1	1	0	0

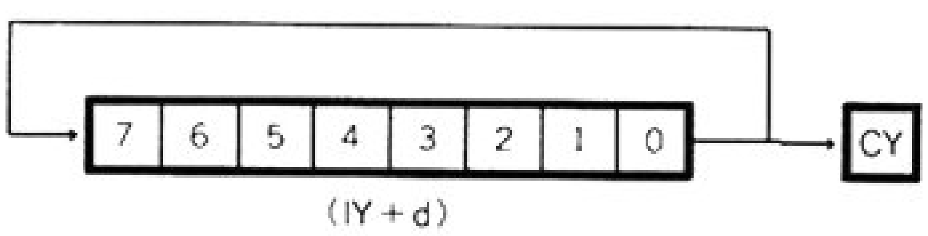
RRC (IY+d)

(Rotate right location (IY+d) circular)　　M＝6　T＝23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	0	0	1	0	1	1	CBH
← d →								
0	0	0	0	1	1	1	0	OEH

【命令の機能】



インデックス・レジスタIYとディスプレースメントdの和で示されるアドレスのデータを1ビット右にシフトする。

データのビット0はキャリー・フラグとデータのビット7に入る。

【例】

IY＝2345H, d＝ABH, (22F0H)＝55Hであるときに、この命令を実行すると、

(22F0H)＝AAH,

C	Z	P/V	S	N	H
1	0	1	1	0	0

RRC r

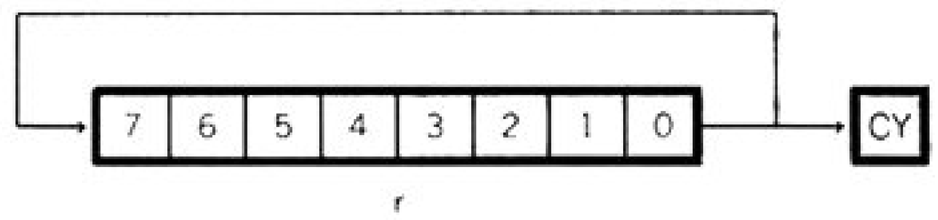
(Rotate right Reg. r circular)　M＝2　T＝8(4, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	0	0	0	1	←—	r	—→	

r	rの値	ニーモニック	オブジェクト・コード
B	0 0 0	RRC　B	0 8 H
C	0 0 1	RRC　C	0 9 H
D	0 1 0	RRC　D	0 A H
E	0 1 1	RRC　E	0 B H
H	1 0 0	RRC　H	0 C H
L	1 0 1	RRC　L	0 D H
A	1 1 1	RRC　A	0 F H

【命令の機能】



レジスタrの内容を1ビット右にシフトする．レジスタrのビット0はキャリー・フラグとレジスタrのビット7に入る．

【例】

Reg.E＝55Hであるときに，この命令を実行すると，

Reg.E＝AAH，

C	Z	P/V	S	N	H
1	0	1	1	0	0

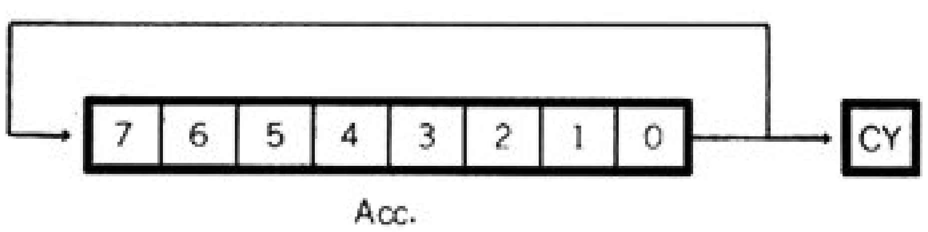
RRCA

(Rotate right Acc. circular)　M＝1　T＝4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	0	0	1	1	1	1	OFH

【命令の機能】



アキュムレータの内容を1ビット右にシフトする．アキュムレータのビット0は，キャリー・フラグとアキュムレータのビット7に入る．

この命令の機能はRRC　r命令において，r＝Aとした場合と同じであるが，この命令のほうが長さが短く，実行速度も速い．ただし，フラグ・レジスタの変化が少し異なる．

【例】

Reg.A＝55Hであるときに，この命令を実行すると，

Reg.A＝AAH，

C	Z	P/V	S	N	H
1	0	1	1	0	0

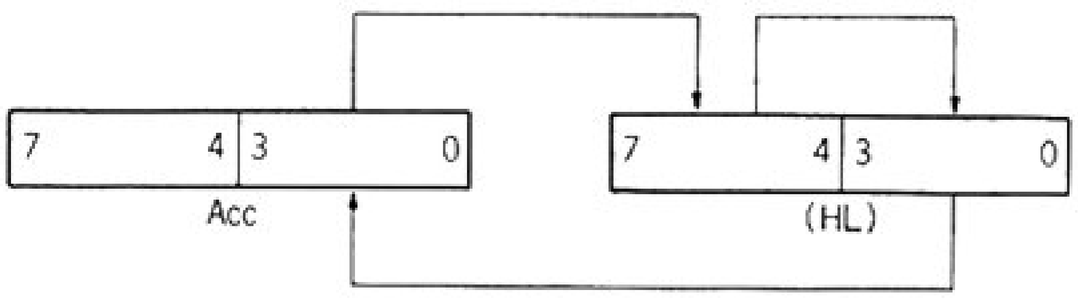
RRD

(Rotate Digit right and left between Acc. and location (HL))　M＝5　T＝18(4, 4, 3, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	1	0	0	1	1	1	67H

【命令の機能】



レジスタ・ペアHLで示されるアドレスのデータの上位4ビットを下位4ビットに移動し，下位4ビットをアキュムレータの下位4ビットに移動する．

アキュムレータの下位4ビットはデータの上位4ビットに移動される．

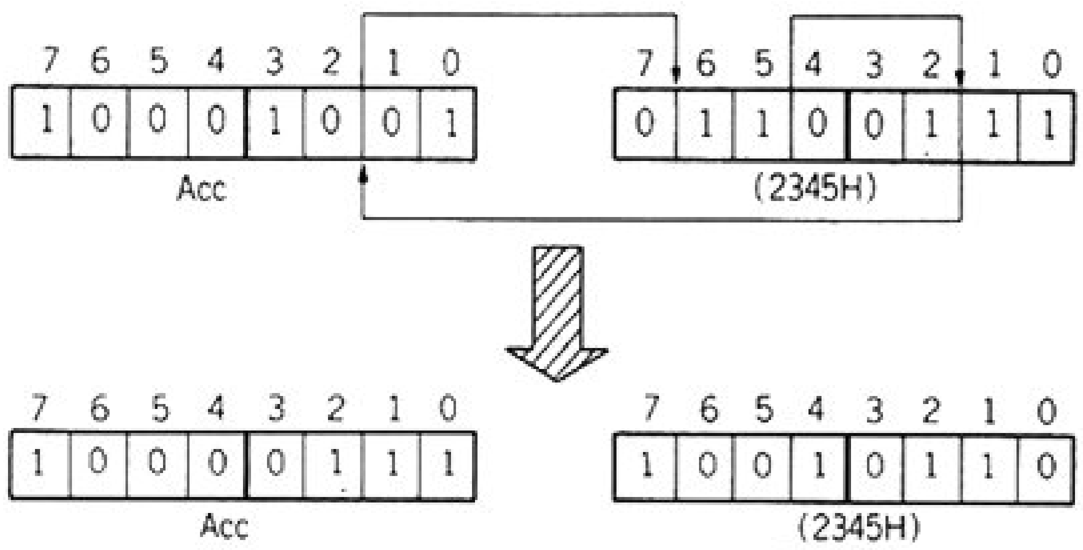
【例】

HL＝2345H，(2345H)＝67H，Reg.A＝89Hであるときに，この命令を実行すると，

(2345H)＝96H，

Reg.A＝87H，

C	Z	P/V	S	N	H
●	0	1	1	0	0



RST n

(Restart to location n)　M＝3　T＝11(5, 3, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0
1	1	←	t	→	1	1	1

n	tの値	ニーモニック	オブジェクト・コード
0 0 H	0 0 0	RST　0 0 H	C 7 H
0 8 H	0 0 1	RST　0 8 H	C F H
1 0 H	0 1 0	RST　1 0 H	D 7 H
1 8 H	0 1 1	RST　1 8 H	D F H
2 0 H	1 0 0	RST　2 0 H	E 7 H
2 8 H	1 0 1	RST　2 8 H	E F H
3 0 H	1 1 0	RST　3 0 H	F 7 H
3 8 H	1 1 1	RST　3 8 H	F F H

【命令の機能】 $\left(\begin{array}{l}(SP-1)\leftarrow PC_H,\ (SP-2)\leftarrow PC_L,\\ PC_H\leftarrow 0,\ PC_L\leftarrow n,\ SP\leftarrow SP-2\end{array}\right)$

スタックにPCの値をセーブした後，n番地へジャンプする．スタック・ポインタSPは2だけ減じられる．フラグ・レジスタの状態は変化しない．

この命令の機能は，00H～38Hに対するCALL命令(CALL　00H～CALL　38H)とまったく同じであるが，CALL命令が3バイト命令であるのに対して，この命令は1バイトである．

モード0および1の割り込みモードでは，この命令がベクトルとして使われる．

【例】

SP＝2000Hであるときに，1000H番地において，RST　28H命令を実行すると，

(1FFFH)＝10H，(1FFEH)＝01H，SP＝1FFEH

となり，0028Hへジャンプする．

SBC A, n

(Subtract n from Acc. with carry)　M＝2　T＝7(4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	1	0	DEH

← n →

【命令の機能】　(A←A－n－CY)

アキュムレータの内容からnとキャリー・フラグの状態(0/1)を引き，結果をアキュムレータにストアする．

【例】

① Reg.A＝55H, CY＝1であるときに，SBC　A, 28H命令を実行すると，

Reg.A＝2CH(55H－28H－1)，

C	Z	P/V	S	N	H
0	0	0	0	1	1

② Reg.A＝55H, CY＝1であるときに，SBC　A, 67H命令を実行すると，

Reg.A＝EDH(55H－67H－1)，

C	Z	P/V	S	N	H
1	0	0	1	1	1

SBC A, (HL)

(Subtract location (HL) from Acc. with carry)　M＝2　T＝7(4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	0	0	1	1	1	1	0	9EH

【命令の機能】　(A←A－(HL)－CY)

アキュムレータの内容から，レジスタ・ペアHLで示されるアドレスのデータとキャリー・フラグの状態(0/1)を引き，結果をアキュムレータにストアする．

【例】

Reg.A＝55H, CY＝1, HL＝2345H, (2345H)＝28Hであるときに，この命令を実行すると，

Reg.A＝2CH(55H－28H－1)，

C	Z	P/V	S	N	H
0	0	0	0	1	1

SBC A, (IX+d)

(Subtract location (IX+d) from Acc. with carry)　M＝5　T＝19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	0	0	1	1	1	1	0	9EH

← d →

【命令の機能】　(A←A－(IX＋d)－CY)

アキュムレータの内容から，インデックス・レジスタIXとディスプレースメントdの和で示されるアドレスのデータとキャリー・フラグの状態(0/1)を引き，結果をアキュムレータにストアする．

【例】

Reg.A＝55H, CY＝1, IX＝2345H, (23ACH)＝28Hであるときに，SBC　A, (IX＋67H)命令を実行すると，

Reg.A＝2CH(55H－28H－1)，

C	Z	P/V	S	N	H
0	0	0	0	1	1

SBC A, (IY+d)

(Subtract location(IY+d) from Acc. with carry)　M=5　T=19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	0	0	1	1	1	1	0	9EH
←			d				→	

【命令の機能】　(A←A－(IY＋d)－CY)

アキュムレータの内容から，インデックス・レジスタ IY とディスプレースメント d の和で示されるアドレスの内容とキャリー・フラグの状態(1/0)を引き，結果をアキュムレータにストアする．

【例】

Reg.A＝55H，CY＝1，IY＝2345H，d＝ABH，(22F0H)＝28H であるときに，この命令を実行すると，

Reg.A＝2CH(55H－28H－1)，

C	Z	P/V	S	N	H
0	0	0	0	1	1

SBC HL, ss

(Subtract Reg.pair ss from HL with carry)　M=4　T=15(4, 4, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	0	1	EDH
0	1	s	s	0	0	1	0	

ss	ssの値	ニーモニック	オブジェクト・コード
BC	0 0	SBC HL, BC	4 2 H
DE	0 1	SBC HL, DE	5 2 H
HL	1 0	SBC HL, HL	6 2 H
SP	1 1	SBC HL, SP	7 2 H

【命令の機能】　(HL←HL－ss－CY)

レジスタ・ペア HL の内容から，レジスタ・ペア ss の内容とキャリー・フラグの状態(1/0)を引き，結果を HL にストアする．

【例】

① HL＝2345H，BC＝1234H，CY＝1 であるときに，SBC HL，BC 命令を実行すると，

HL＝1110H，

C	Z	P/V	S	N	H
0	0	0	0	1	0

② HL＝2345H，DE＝4567H，CY＝1 であるときに，SBC HL，DE 命令を実行すると，

HL＝DDDDH，

C	Z	P/V	S	N	H
1	0	0	1	1	1

SBC A, r

(Subtract Reg. r from Acc. with carry)　M=1　T=4

【オブジェクト・コード】

7	6	5	4	3	2	1	0
1	0	0	1	1	←	r	→

r	rの値	ニーモニック	オブジェクト・コード
B	0 0 0	SBC A, B	9 8 H
C	0 0 1	SBC A, C	9 9 H
D	0 1 0	SBC A, D	9 A H
E	0 1 1	SBC A, E	9 B H
H	1 0 0	SBC A, H	9 C H
L	1 0 1	SBC A, L	9 D H
A	1 1 1	SBC A, A	9 F H

【命令の機能】　(A←A－r－CY)

アキュムレータの内容から，レジスタ r とキャリー・フラグの状態(1/0)を引き，結果をアキュムレータにストアする．

【例】

① Reg.A＝55H，CY＝1，Reg.B＝28H であるときに，SBC A，B 命令を実行すると，

Reg.A＝2CH，

C	Z	P/V	S	N	H
0	0	0	0	1	1

② Reg.A＝55H，CY＝1，Reg.L＝67H であるときに，SBC A，L 命令を実行すると，

Reg.A＝EDH，

C	Z	P/V	S	N	H
1	0	0	1	1	1

SCF

(Set carry flag)　M=1　T=4

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
0	0	1	1	0	1	1	1	37H

【命令の機能】　(CY←1)

フラグ・レジスタ中のキャリー・フラグをセット(1)する．

SET b, (HL)

(Set bit b of location (HL))　M=4　T=15(4, 4, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
1	1	b			1	1	0	

b	bの値	ニーモニック	オブジェクト・コード
0	0 0 0	SET 0, (HL)	C 6 H
1	0 0 1	SET 1, (HL)	C E H
2	0 1 0	SET 2, (HL)	D 6 H
3	0 1 1	SET 3, (HL)	D E H
4	1 0 0	SET 4, (HL)	E 6 H
5	1 0 1	SET 5, (HL)	E E H
6	1 1 0	SET 6, (HL)	F 6 H
7	1 1 1	SET 7, (HL)	F E H

【命令の機能】　$((HL)_b \leftarrow 1)$

レジスタ・ペアHLで示されるアドレスのデータのビットbを1にする．

フラグ・レジスタの状態は変化しない．

【例】

HL＝2345H，(2345H)＝55Hであるときに，SET 7，(HL)命令を実行すると，
(2345H)＝D5Hとなる．

SET b, (IX+d)

(Set bit b of location (IX+d))　M=6　T=23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	0	0	1	0	1	1	CBH
←d→								
1	1	←b→			1	1	0	

b	bの値	ニーモニック	オブジェクト・コード
0	0 0 0	SET 0, (IX+d)	C 6 H
1	0 0 1	SET 1, (IX+d)	C E H
2	0 1 0	SET 2, (IX+d)	D 6 H
3	0 1 1	SET 3, (IX+d)	D E H
4	1 0 0	SET 4, (IX+d)	E 6 H
5	1 0 1	SET 5, (IX+d)	E E H
6	1 1 0	SET 6, (IX+d)	F 6 H
7	1 1 1	SET 7, (IX+d)	F E H

【命令の機能】　$((IX+d)_b \leftarrow 1)$

インデックス・レジスタIXとディスプレースメントdの和で示されるアドレスのデータのビットbを1にする．

フラグ・レジスタの状態は変化しない．

【例】

IX＝2345H，d＝67H，(23ACH)＝55Hであるときに，SET 1，(IX＋d)命令を実行すると，
(23ACH)＝57Hとなる．

SET b, (IY+d) (Set bit b of location (IY+d))

M=6
T=23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	0	0	1	0	1	1	CBH
← d →								
1	1	← b →			1	1	0	

b	bの値	ニーモニック	オブジェクト・コード
0	0 0 0	SET b, (IY+d)	C 6 H
1	0 0 1	SET b, (IY+d)	C E H
2	0 1 0	SET b, (IY+d)	D 6 H
3	0 1 1	SET b, (IY+d)	D E H
4	1 0 0	SET b, (IY+d)	E 6 H
5	1 0 1	SET b, (IY+d)	E E H
6	1 1 0	SET b, (IY+d)	F 6 H
7	1 1 1	SET b, (IY+d)	F E H

【命令の機能】 $((IY+d)_b \leftarrow 1)$

インデックス・レジスタIYとディスプレースメントdの和で示されるアドレスのデータのビットbを1にする.

フラグ・レジスタの状態は変化しない.

【例】

IY=2345H, d=ABH, (22F0H)=55Hであるときに, SET 5, (IY+d)命令を実行すると,
(22F0H)=75Hとなる.

SET b, r (Set bit b of Reg. r)

M=2
T=8(4, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
1	1	← b →			← r →			

		レジスタr						
		B	C	D	E	H	L	A
ビットb	0	C0	C1	C2	C3	C4	C5	C7
	1	C8	C9	CA	CB	CC	CD	CF
	2	D0	D1	D2	D3	D4	D5	D7
	3	D8	D9	DA	DB	DC	DD	DF
	4	E0	E1	E2	E3	E4	E5	E7
	5	E8	E9	EA	EB	EC	ED	EF
	6	F0	F1	F2	F3	F4	F5	F7
	7	F8	F9	FA	FB	FC	FD	FF

【命令の機能】 $(r_b \leftarrow 1)$

レジスタrのビットbを1にする. フラグ・レジスタの状態は変化しない.

【例】

Reg.B=55Hであるときに, SET 1, B命令を実行すると,
Reg.B=57Hとなる.

SLA (HL)

(Shift location (HL) left arithmetic)　M＝4　T＝15(4, 4, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	0	1	0	0	1	1	0	26H

【命令の機能】

CY ← | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ← 0

(HL)

レジスタ・ペア HL で示されるアドレスのデータを，1ビット左にシフトする．データのビット7はキャリー・フラグに入り，データのビット0にはゼロが入る．

【例】

HL＝2345H，(2345H)＝55H であるときに，この命令を実行すると，

(2345H)＝AAH，

C	Z	P/V	S	N	H
0	0	1	1	0	0

SLA (IX+d)

(Shift location (IX+d) left arithmetic)　M＝6　T＝23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	0	0	1	0	1	1	CBH
←			d				→	
0	0	1	0	0	1	1	0	26H

【命令の機能】

CY ← | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ← 0

(IX＋d)

インデックス・レジスタ IX とディスプレースメント d の和で示されるアドレスのデータを，1ビット左にシフトする．データのビット7はキャリー・フラグに入り，ビット0にはゼロが入る．

【例】

IX＝2345H，d＝67H，(23ACH)＝AAH であるときに，この命令を実行すると，

(23ACH)＝54H，

C	Z	P/V	S	N	H
1	0	0	0	0	0

SLA (IY+d)

(Shift location (IY+d) left arithmetic)　M＝6　T＝23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	0	0	1	0	1	1	CBH
←			d				→	
0	0	1	0	0	1	1	0	26H

【命令の機能】

CY ← | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ← 0

(IY＋d)

インデックス・レジスタ IY とディスプレースメント d の和で示されるアドレスのデータを1ビット左にシフトする．

データのビット7はキャリー・フラグに入り，ビット0にはゼロが入る．

【例】

IY＝2345H，d＝ABH，(22F0H)＝AAH であるときに，この命令を実行すると，

(22F0H)＝54H，

C	Z	P/V	S	N	H
1	0	0	0	0	0

SLA r

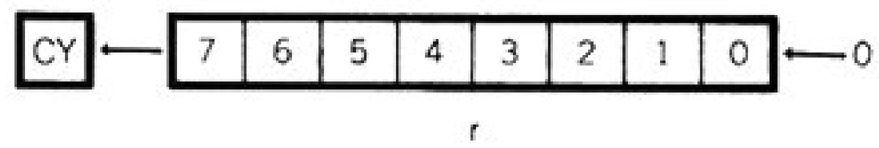
(Shift Reg. r left arithmetic) M＝2 T＝8(4, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	0	1	0	0	←r→			

r	rの値	ニーモニック	オブジェクト・コード
B	000	SLA B	20H
C	001	SLA C	21H
D	010	SLA D	22H
E	011	SLA E	23H
H	100	SLA H	24H
L	101	SLA L	25H
A	111	SLA A	27H

【命令の機能】



レジスタrの内容を1ビット左にシフトする．レジスタrのビット7はキャリー・フラグに入り，ビット0にはゼロが入る．

【例】

Reg.E＝AAHであるときに，SLA　E命令を実行すると，

Reg.E＝54H，

C	Z	P/V	S	N	H
1	0	0	0	0	0

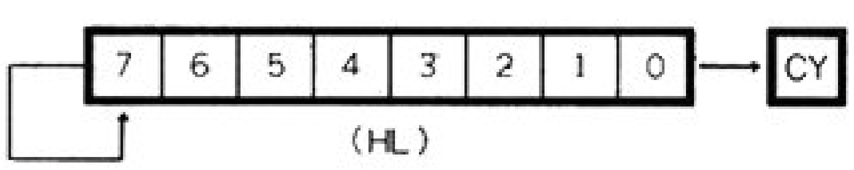
SRA (HL)

(Shift location (HL) right arithmetic) M＝4 T＝15(4, 4, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	0	1	0	1	1	1	0	2EH

【命令の機能】



レジスタ・ペアHLで示されるアドレスのデータを，1ビット右にシフトする．

データのビット0はキャリー・フラグに入り，ビット7の状態は変化しない．

【例】

HL＝2345H，(2345H)＝A5Hであるときに，この命令を実行すると，

(2345H)＝D2H，

C	Z	P/V	S	N	H
1	0	1	1	0	0

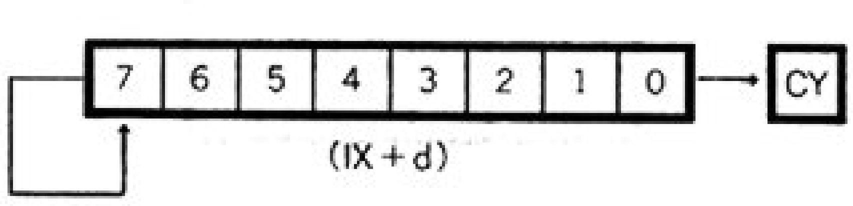
SRA (IX+d)

(Shift location (IX+d) right arithmetic) M＝6 T＝23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	0	0	1	0	1	1	CBH
←d→								
0	0	1	0	1	1	1	0	2EH

【命令の機能】



インデックス・レジスタIXとディスプレースメントdの和で示されるアドレスのデータを1ビット右にシフトする．データのビット0はキャリー・フラグに入り，ビット7の状態は変らない．

【例】

IX＝2345H，d＝67H，(23ACH)＝A5Hであるときに，この命令を実行すると，

(23ACH)＝D2H，

C	Z	P/V	S	N	H
1	0	1	1	0	0

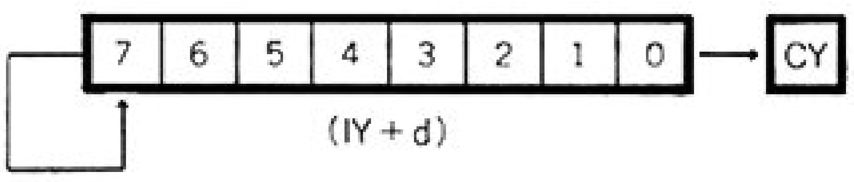
SRA (IY+d)

(Shift location (IY+d) right arithmetic)　M=6　T=23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	0	0	1	0	1	1	CBH
←— d —→								
0	0	1	0	1	1	1	0	2EH

【命令の機能】



インデックス・レジスタIYとディスプレースメントdの和で示されるアドレスのデータを1ビット右にシフトする．

データのビット0はキャリー・フラグに入り，ビット7の状態は変らない．

【例】

IY=2345H，d=ABH，(22F0H)=A5Hであるときに，この命令を実行すると，

(22F0H)=D2H，

C	Z	P/V	S	N	H
1	0	1	1	0	0

SRA r

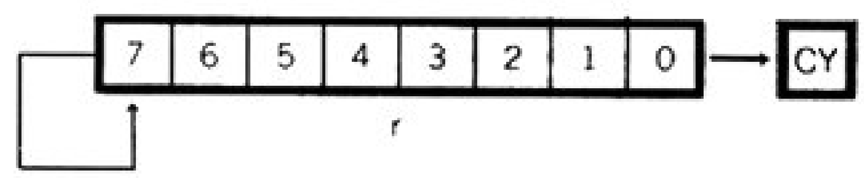
(Shift Reg. r right arithmetic)　M=2　T=8(4, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	0	1	0	1	←— r —→			

r	rの値	ニーモニック	オブジェクト・コード
B	0 0 0	SRA B	2 8 H
C	0 0 1	SRA C	2 9 H
D	0 1 0	SRA D	2 A H
E	0 1 1	SRA E	2 B H
H	1 0 0	SRA H	2 C H
L	1 0 1	SRA L	2 D H
A	1 1 1	SRA A	2 F H

【命令の機能】



レジスタrの内容を1ビット右にシフトする．レジスタrのビット0はキャリー・フラグに入り，ビット7の状態は変らない．

【例】

Reg.E=A5Hであるとき，SRA　E命令を実行すると，

Reg.E=D2H，

C	Z	P/V	S	N	H
1	0	1	1	0	0

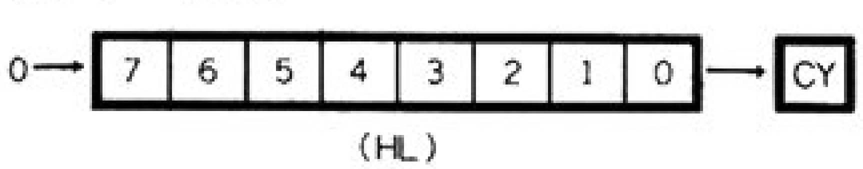
SRL (HL)

(Shift location (HL) right logical)　M=4　T=15(4, 4, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	0	1	1	1	1	1	0	3EH

【命令の機能】



レジスタ・ペアHLで示されるアドレスのデータを，1ビット右にシフトする．データのビット0はキャリー・フラグに入り，ビット7にはゼロが入る．

【例】

HL=2345H，(2345H)=A5Hであるときに，この命令を実行すると，

(2345H)=52H，

C	Z	P/V	S	N	H
1	0	0	0	0	0

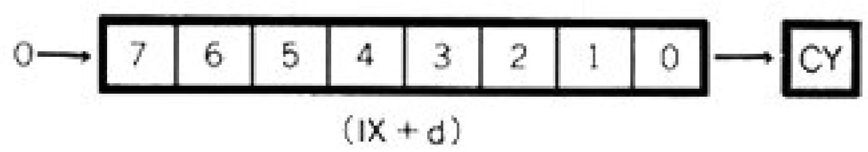
SRL (IX+d)

(Shift location (IX+d) right logical)　M=6　T=23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	1	0	0	1	0	1	1	CBH
← d →								
0	0	1	1	1	1	1	0	3EH

【命令の機能】



インデックス・レジスタ IX とディスプレースメント d の和で示されるアドレスのデータを 1 ビット右にシフトする．データのビット 0 はキャリー・フラグに入り，ビット 7 には 0 が入る．

【例】

IX=2345H，d=67H，(23ACH)=A5H であるときに，この命令を実行すると，

(23ACH)=52H，

C	Z	P/V	S	N	H
1	0	0	0	0	0

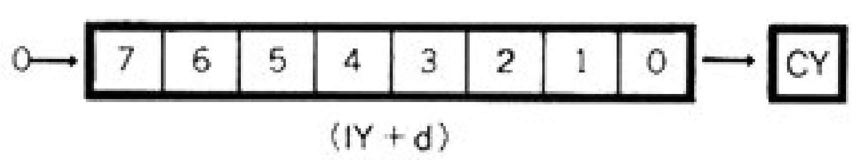
SRL (IY+d)

(Shift location (IY+d) right logical)　M=6　T=23(4, 4, 3, 5, 4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	1	0	0	1	0	1	1	CBH
← d →								
0	0	1	1	1	1	1	0	3EH

【命令の機能】



インデックス・レジスタ IY とディスプレースメント d の和で示されるアドレスのデータを 1 ビット右にシフトする．

データのビット 0 はキャリー・フラグに入り，ビット 7 にはゼロが入る．

【例】

IY=2345H，d=ABH，(22F0H)=A5H であるときに，この命令を実行すると，

(22F0H)=52H，

C	Z	P/V	S	N	H
1	0	0	0	0	0

SRL r

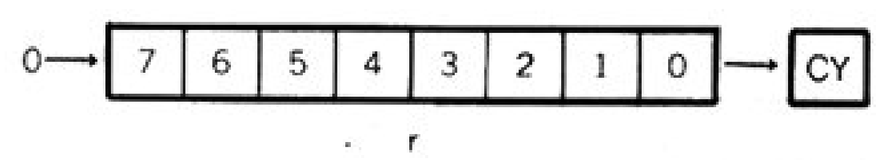
(Shift Reg. r right logical)　M=2　T=8(4, 4)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	0	1	0	1	1	CBH
0	0	1	1	1	← r →			

r	r の値	ニーモニック	オブジェクト・コード
B	000	SRL B	38H
C	001	SRL C	39H
D	010	SRL D	3AH
E	011	SRL E	3BH
H	100	SRL H	3CH
L	101	SRL L	3DH
A	111	SRL A	3FH

【命令の機能】



レジスタ r の内容を 1 ビット右にシフトする．レジスタ r のビット 0 はキャリー・フラグに入り，ビット 7 にはゼロが入る．

【例】

Reg.L=A5H であるときに，SRL L 命令を実行すると，

Reg.L=52H，

C	Z	P/V	S	N	H
1	0	0	0	0	0

SUB (HL)

(Subtract location (HL) from Acc.)　M=2　T=7(4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	0	0	1	0	1	1	0	96H

【命令の機能】　(A←A－(HL))

アキュムレータから，レジスタ・ペアHLで示されるアドレスのデータを引き，結果をアキュムレータにストアする．

【例】

HL＝2345H，(2345H)＝55H，Reg.A＝78Hであるとき，この命令を実行すると，

Reg.A＝23H，

C	Z	P/V	S	N	H
0	0	1	0	1	0

SUB (IX+d)

(Subtract location (IX+d) from Acc.)　M=5　T=19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	0	0	1	0	1	1	0	96H
←─			d				─→	

【命令の機能】　(A←A－(IX＋d))

インデックス・レジスタIXとディスプレースメントdの和で示されるアドレスのデータを，アキュムレータから引き，結果をアキュムレータにストアする．

【例】

Reg.A＝55H，IX＝2345H，d＝67H，(23ACH)＝78Hであるときに，この命令を実行すると，

Reg.A＝DDH，

C	Z	P/V	S	N	H
1	0	0	1	1	1

SUB (IY+d)

(Subtract location (IY+d) from Acc.)　M=5　T=19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	0	0	1	0	1	1	0	96H
←─			d				─→	

【命令の機能】　(A←A－(IY＋d))

インデックス・レジスタIYとディスプレースメントdの和で示されるアドレスのデータを，アキュムレータから引き，結果をアキュムレータにストアする．

【例】

Reg.A＝55H，IY＝2345H，d＝ABH，(22F0H)＝78Hであるときに，この命令を実行すると，

Reg.A＝DDH，

C	Z	P/V	S	N	H
1	0	0	1	1	1

SUB r

M=1
(Subtract Reg. r from Acc.) T=4

【オブジェクト・コード】

7	6	5	4	3	2	1	0
1	0	0	1	0	←r→		

r	rの値	ニーモニック	オブジェクト・コード
B	0 0 0	SUB B	9 0 H
C	0 0 1	SUB C	9 1 H
D	0 1 0	SUB D	9 2 H
E	0 1 1	SUB E	9 3 H
H	1 0 0	SUB H	9 4 H
L	1 0 1	SUB L	9 5 H
A	1 1 1	SUB A	9 7 H

【命令の機能】 (A←A－r)

アキュムレータからレジスタ r の内容を引き，結果をアキュムレータにストアする．

【例】

① Reg.A＝55H，Reg.D＝78H であるとき，SUB D 命令を実行すると，

Reg.A＝DDH，

C	Z	P/V	S	N	H
1	0	0	1	1	1

② Reg.A＝XXH(何でもよい)であるときに，SUB A 命令を実行すると，

Reg.A＝00H，

C	Z	P/V	S	N	H
0	1	0	0	1	0

SUB n

M＝2
(Subtract n from Acc.) T＝7(4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	0	1	1	0	D6H
←n→								

【命令の機能】 (A←A－n)

アキュムレータからイミーディエイト・データ n を引き，結果をアキュムレータにストアする．

【例】

Reg.A＝55H であるときに，SUB 23H 命令を実行すると，

Reg.A＝32H，

C	Z	P/V	S	N	H
0	0	0	0	1	0

XOR (HL)

M＝2
(Exclusive OR location (HL) and Acc.) T＝7(4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	0	1	0	1	1	1	0	AEH

【命令の機能】 (A←A ⊕(HL))

レジスタ・ペア HL で示されるアドレスのデータと，アキュムレータの内容との排他論理和をとり，結果をアキュムレータにストアする．

【例】

Reg.A＝55H，HL＝2345H，(2345H)＝5AH であるとき，この命令を実行すると，

Reg.A＝0FH，

C	Z	P/V	S	N	H
0	0	1	0	0	0

XOR (IX+d)

(Exclusive OR location (IX+d) and Acc.)

M=5
T=19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	0	1	1	1	0	1	DDH
1	0	1	0	1	1	1	0	AEH
←d→								

【命令の機能】　(A←A ⊕(IX+d))

インデックス・レジスタ IX とディスプレースメント d の和で示されるアドレスのデータとアキュムレータの内容との排他論理和をとり，結果をアキュムレータにストアする．

【例】

Reg.A=55H, IX=2345H, d=67H, (23ACH)=A5H であるときに，この命令を実行すると，

Reg.A=F0H,

C	Z	P/V	S	N	H
0	0	1	1	0	0

XOR (IY+d)

(Exclusive OR location (IY+d) and Acc.)

M=5
T=19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	FDH
1	0	1	0	1	1	1	0	AEH
←d→								

【命令の機能】　(A←A ⊕(IY+d))

インデックス・レジスタ IY とディスプレースメント d の和で示されるアドレスのデータと，アキュムレータの内容との排他論理和をとり，結果をアキュムレータにストアする．

【例】

Reg.A=A5H, IY=2345H, d=ABH, (22F0H)=48H であるときに，この命令を実行すると，

Reg.A=EDH,

C	Z	P/V	S	N	H
0	0	1	1	0	0

XOR r

(Exclusive OR Reg. r and Acc.)

M=5
T=19(4, 4, 3, 5, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0
1	0	1	0	1	←r→		

r	rの値	ニーモニック	オブジェクト・コード
B	000	XOR B	A8H
C	001	XOR C	A9H
D	010	XOR D	AAH
E	011	XOR E	ABH
H	100	XOR H	ACH
L	101	XOR L	ADH
A	111	XOR A	AFH

【命令の機能】　(A←A ⊕r)

アキュムレータの内容とレジスタ r の内容との排他論理和をとり，結果をアキュムレータにストアする．

【例】

① Reg.A=46H, Reg.D=D4H であるときに，XOR D 命令を実行すると，

Reg.A=92H,

C	Z	P/V	S	N	H
0	0	0	1	0	0

② Reg.A=XXH(何でもよい)であるときに，XOR A 命令を実行すると，

Reg.A=00H,

C	Z	P/V	S	N	H
0	1	1	0	0	0

XOR n

(Exclusive OR n and Acc.) M=2 T=7(4, 3)

【オブジェクト・コード】

7	6	5	4	3	2	1	0	
1	1	1	0	1	1	1	0	EEH

←— n —→

【命令の機能】 (A←A⊕n)

アキュムレータの内容とイミーディエイト・データnとの排他論理和をとり，結果をアキュムレータにストアする．

【例】

Reg.A＝55Hであるときに，XOR AAH命令を実行すると，

Reg.A＝FFH，

C	Z	P/V	S	N	H
0	0	1	1	0	0

Z 80のプログラム例

ここではZ 80のプログラムの中に必ず現われるといっても過言ではない，基本的なサブルーチンを紹介します．

■ブロック移動(Block Move) (Prog❶)

長さ4096バイトのブロック・データをメモリ内で移動するプログラムです．このプログラムはSRCEという名のバッファのデータ，4096バイトをDESTという名のバッファに移動します．

■メモリ・クリヤ(Memory Clear) (Prog❷)

長さ4096バイトのバッファ(BUFFER)を，アキュムレータの内容によりクリヤするプログラムです．このプログラムの動作原理はブロック移動で，まずバッファの先頭をアキュムレータのデータでクリヤしておき，次にLDIR命令を用いてバッファ内全体を先頭のデータで満たします．図24にこのプログラムの動作を示します．

■タイミング・ループ (Prog❸)

タイミング・ループ・プログラムの例です．このプログラムはレジスタ・ペアBCの値を1ずつ減じ，ゼロになるまで同一ループを繰り返します．このサブ・ルーチンに入ってから抜け出るまでの時間Tは，

$T＝(BC－1)×26＋31$ (ステート)

となります．したがって，Z 80 Aが4 MHzで動作しているものとすると，BC＝10000_{10}であるときの時間は，

$(10000－1)×26＋31＝260.005$ (ステート)

約65 msとなります．

■ストリング・データの比較 (Prog❹)

二つのストリング・データを比較するサブルーチンです．二つのデータの先頭アドレスをレジスタ・ペアDE，およびHLにセットし，データの長さをレジスタCにセットして，このサブルーチンをコールします．

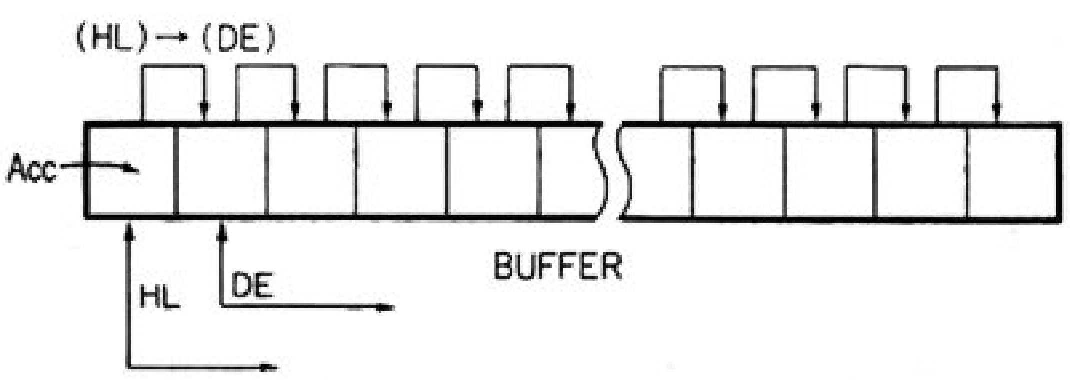
比較結果はフラグ・レジスタにセットされますが，両者が等しいときにはゼロ・フラグがセットされ，レジスタ・ペアDEで示すデータのほうが大きいときには，キャリー・フラグがゼロになり，その反対である場合はキャリー・フラグが"1"になります．

■ブロック・データの左回転 (Prog❺)

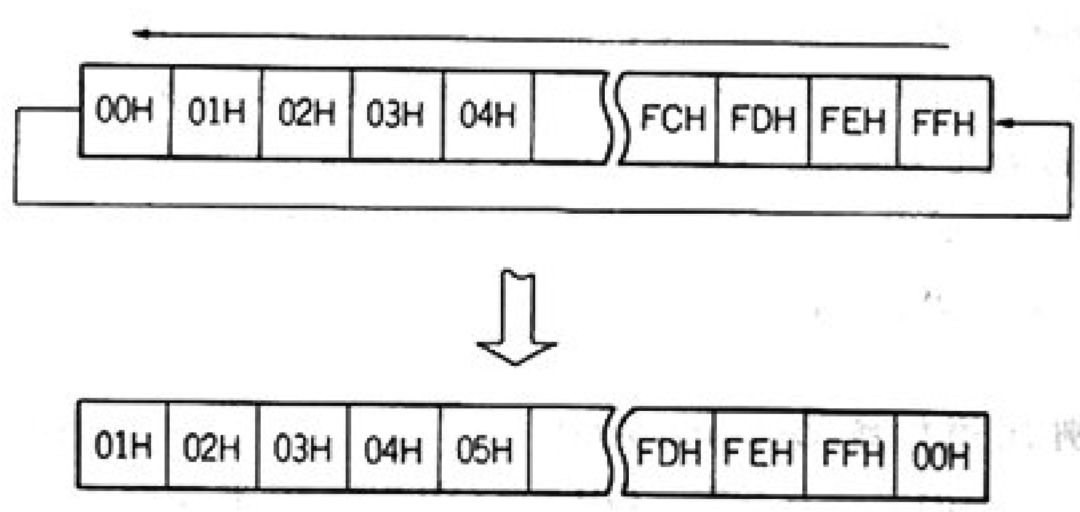
ブロック・データを左回転するサブルーチンです．データの先頭アドレスをレジスタ・ペアHLにセットし，データの長さをBCにセットして，このサブルーチンをコールします．

このプログラムの動作を図25に示します．

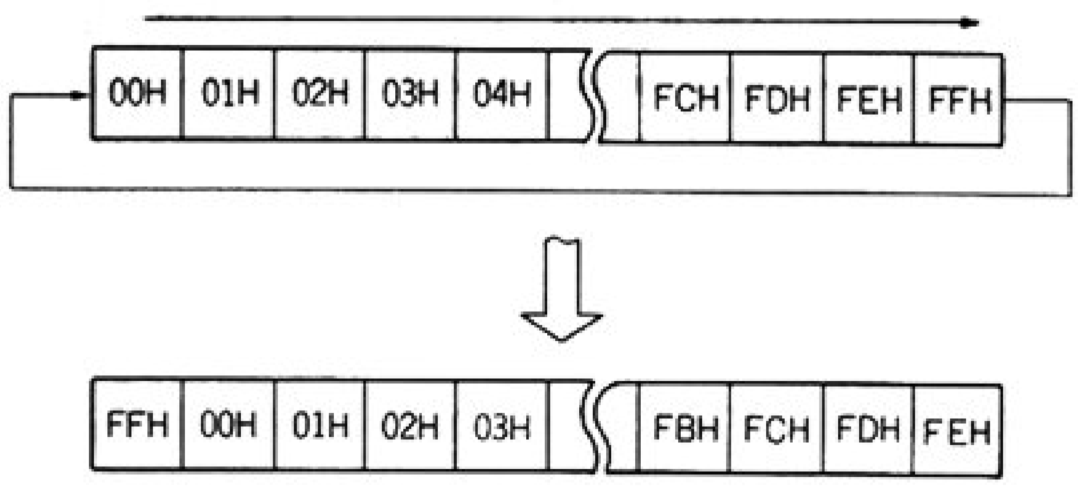
〈図24〉 Prog❷の動作．LDIR命令により(HL)→(DE)の操作を4095回繰り返す



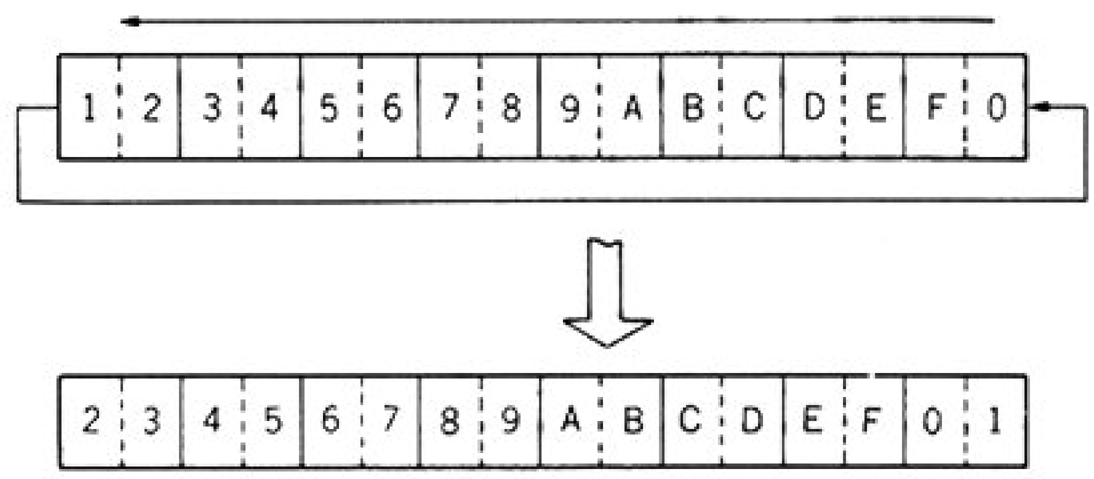
〈図25〉 Prog❺の動作



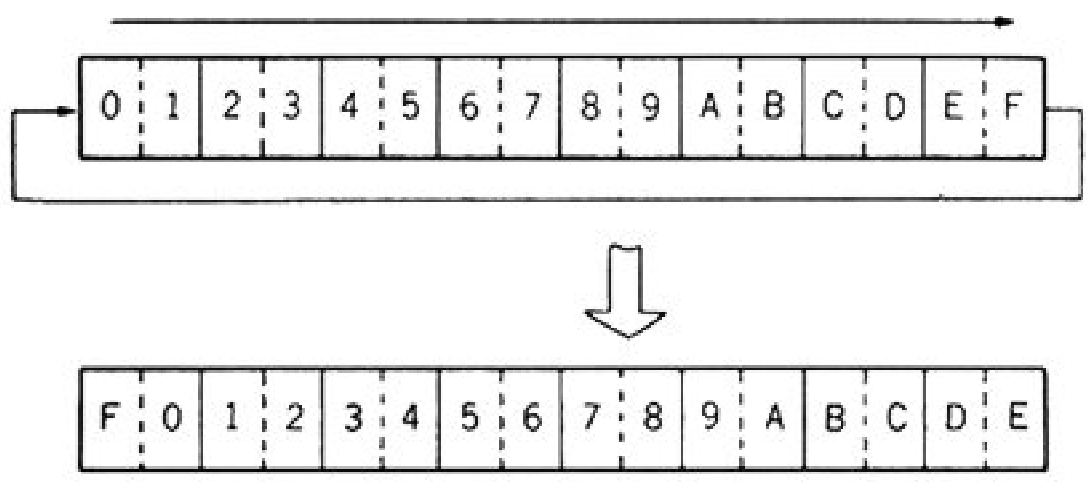
〈図26〉 Prog❻の動作



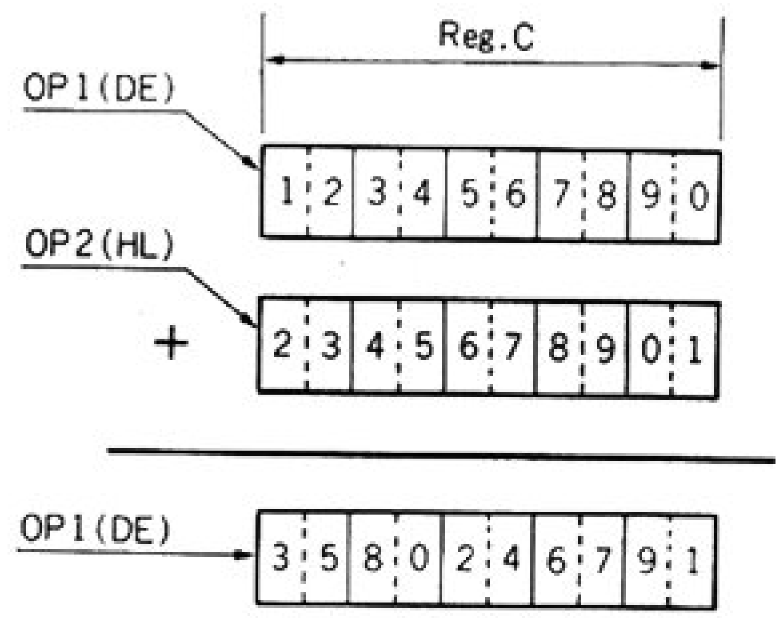
〈図27〉 Prog❼の動作



〈図28〉 Prog❽の動作



〈図29〉
Prog❾の動作



■ブロック・データの右回転　(Prog❻)

ブロック・データを右回転するサブルーチンです．データの先頭アドレスを，レジスタ・ペアHLにセットし，データの長さをBCにセットして，このサブルーチンをコールします．このプログラムの動作を，図26に示します．

■ブロック・データの
ディジット単位の左回転(Prog❼)

ブロック・データをディジット単位(4ビット)で左回転するプログラムです．データの先頭アドレスをレジスタ・ペアHLにセットし，データの長さをレジスタCにセットし，このサブルーチンをコールします．

このプログラムはRLD命令の使いかたを学ぶよい例になると思います．図27にこのプログラムの動作を示します．

■ブロック・データの
ディジット単位の右回転(Prog❽)

ブロック・データをディジット(4ビット)単位で右回転するプログラムで，これはRRD命令の使用法を示しています．データの先頭アドレスをレジスタ・ペアHLにセットし，データの長さをレジスタCにセットして，このサブ・ルーチンをコールします．このプログラムの動作を図28に示します．

■10進加算(Decimal Add)　(Prog❾)

二つのオペランド，OP1とOP2の10進和を求め，結果をOP1にストアするサブルーチンです．レジスタ・ペアDEにOP1のアドレス，HLにOP2のアドレスをセットし，レジスタCにオペランドの長さをセットして，このサブルーチンをコールします．この例では両オペランドの長さは等しいものと仮定します．

図29にこのプログラムの動作を示します．オーバフローが起こると，サブルーチンを抜け出すときにキャリー・フラグがセットされます．

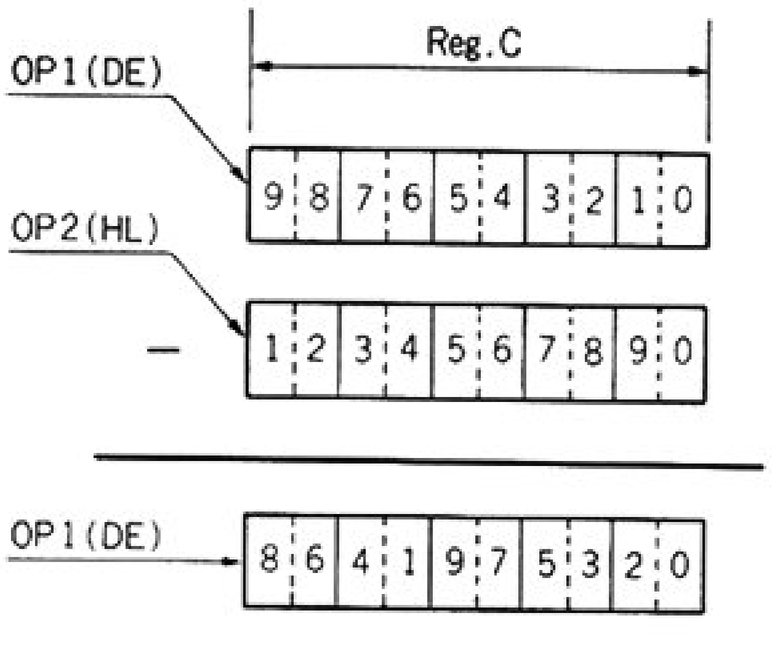
■10進減算(Decimal Subtract)　(Prog❿)

二つのオペランド，OP1とOP2の10進差を求め，結果をOP1にストアするプログラムです．OP1およびOP2のアドレスを，それぞれレジスタ・ペアDE，およびHLにセットし，オペランドの長さをレジスタCにセットし，このサブ・ルーチンをコールします．

このプログラムではOP1とOP2の長さは等しく，かつOP1≧OP2であると仮定しています．このプログラムの動作を図30に示します．このプログラムと前出のProg❾とを比較してみれば明らかなように，異なるのはADC命令とSBC命令だけです．

■テーブル・ジャンプ(Table Jump)　(Prog⓫)

レジスタLの値にしたがって，異なるルーチンにジャンプするプログラムです．レジスタLの値が00Hであると，ROUT00へジャンプし，01HであるとROUT01へジャンプします．またレジスタLの値がFEHであるとROUTFE，FFHであるとROUTFFへジャンプします．



<図30>
Prog❿の動作

このプログラムは，レジスタLの値を2倍し，それに各ルーチンのアドレス・テーブルのベース・アドレス(TABLE)を加え，各ルーチンのアドレスを求めジャンプします．

■コード変換(Translate)　(Prog⓬)

コード変換を行うサブ・ルーチンです．このプログラムはレジスタCのデータを，コード変換テーブルを用いてコード変換をします．レジスタCの値と変換テーブルのベース・アドレス(TABLE)を加えて得られたアドレスのデータを，アキュムレータにロードします．アキュムレータのデータが変換されたデータです．

■コード・デコーダ(Code Decoder)　(Prog⓭)

アキュムレータに入っているデータの内容をチェックし，それがレパートリにある場合にのみ，データごとに異なるルーチンへジャンプします．このプログラムにおけるコードのレパートリは，A，B，P，S，Y，Zです．

アキュムレータのデータがこれらのコードと一致すると，ROUTA～ROUTZへジャンプします．もしレパートリにない場合には，ゼロ・フラグをクリヤしてリターンします．

■ASCII→16進変換　(Prog⓮)

ASCII文字データを16進数に変換するプログラムです．このプログラムは文字データ0(30H)～9(39H)を00H～09Hに変換し，A(41H)～F(46H)を0AH～0FHに変換します．

0～9の場合には上位4ビットをゼロにすればすむのですが，A～Fの場合には9を加えて4AH～4FHにした後，上位4ビットをゼロにします．なお，このサブ・ルーチンは文字データが0～9，A～F以外である場合は，キャリー・フラグをセットしてリターンします．

■16進→ASCII変換　(Prog⓯)

16進数をASCII文字に変換するプログラムです．このサブ・ルーチンは，アキュムレータ中の16進数データを2バイトのASCII文字に変換し，レジスタB，およびレジスタCにストアします．

いまアキュムレータのデータが5FHであったとすると，レジスタBには5(35H)，レジスタCにはF(46H)がストアされます．変換はアキュムレータの上位4ビットと下位4ビットに分けて行われ，各4ビットの値が0～9である場合には30Hを加え，A～Fである場合には，いったん7を加えた後に30Hを加えます．

■ストリング・データの表示　(Prog⓰)

ストリング・データをTTYやCRTに表示する例を二つ示します．このプログラム中のCOUTというサブ・ルーチンは，無定義ラベルのエラー表示(U)がされています．このルーチンはアキュムレータの内容をTTYやCRTに出力するルーチンであり，個々のハードウェアに依存しますので，あえて無定義のままにしてあります．

このプログラムには同じストリング・データ(THIS IS AN MESSAGE)を表示する二つのサブ・ルーチン(DISPとDISPX)が示されています．どちらの場合においても，ストリング・データの終りを示す，エンド・センティネル(Sentinel：見張り)はコード0FFHです．

サブ・ルーチンDISPは，レジスタ・ペアHLで示されるアドレスのデータをストリング・データとして取扱います．サブ・ルーチンDISPXはCALL命令の直後から続くデータをストリング・データとして取り扱います．

DISPXの方法は，ストリング・データのラベルを定義する必要もありませんし，レジスタHLにストリング・データのアドレスをロードする必要もないので，わりと便利なものです．

〈Prog❶〉 ブロック移動

```
                        ;*************************************************
                        ;****           BLOCK   MOVE                 ****
                        ;*************************************************
0000    21 000B         BMOVE:  LD      HL,SRCE         ;LOAD SRCE ADDRESS
0003    11 100B                 LD      DE,DEST         ;LOAD DEST ADDRESS
0006    01 0010                 LD      BC,LENGTH       ;LOAD LENGTH TO MOVE
0009    ED B0                   LDIR                    ;MOVE DATA
                        ;
                        ;
000B                    SRCE:   DS      4096            ;SOURCE DATA
100B                    DEST:   DS      4096            ;DESTINATION BUFFER
1000                    LENGTH  EQU     DEST-SRCE       ;SRCE LENGTH
                        ;
                                END
```

〈Prog❷〉 メモリ・クリヤ．アキュムレータの内容でバッファをクリヤする

```
                        ;*************************************************
                        ;****           MEMORY CLEAR                 ****
                        ;*************************************************
                        ;       A       DATA TO BE FILLED
0000    21 0C00         CLEAR:  LD      HL,BUFFER       ;LOAD BUFFER ADDRESS
0003    01 FF0F                 LD      BC,LENGTH-1     ;LOAD BUFFER LENGTH
0006    77                      LD      (HL),A          ;CLEAR TOP OF BUFFER
0007    11 0D00                 LD      DE,BUFFER+1     ;LOAD BUFFER ADDRESS
000A    ED B0                   LDIR                    ;FILL BUFFER WITH A
                        ;
000C                    BUFFER: DS      4096            ;
1000                    LENGTH  EQU     $-BUFFER        ;
                        ;
                                END
```

〈Prog❸〉 タイミング・ループ

```
                        ;*************************************************
                        ;****           TIMING LOOP                  ****
                        ;*************************************************
                        ;       BC      LOOP COUNTER
                        ;       TIME=(BC-1)*26+31 STATES
0000    0B              TIME:   DEC     BC              ;COUNT DOWN
0001    78                      LD      A,B             ;COUNT EXPIRED ?
0002    B1                      OR      C               ;
0003    20 FB                   JR      NZ,TIME         ;NO,TIME
0005    C9                      RET                     ;
                        ;
                                END
```

〈Prog❹〉 ストリング・データの比較

```
                        ;*************************************************
                        ;****           COMPARE STRING  DATA         ****
                        ;*************************************************
                        ;       DE      OPERAND 1
                        ;       HL      OPERAND 2
                        ;       B       LENGTH TO COMPARE
0000    1A              COMP:   LD      A,(DE)          ;LOAD OPERAND 1
0001    BE                      CP      (HL)            ;COMPARE IT WITH OP2
0002    C0                      RET     NZ              ;RETURN UNLESS MATCH
0003    13                      INC     DE              ;BUMP OP1 POINTER
0004    23                      INC     HL              ;BUMP OP2 POINTER
0005    10 F9                   DJNZ    COMP            ;COMPARED ALL ?
                                                        ;GO TO COMP IF NOT YET
0007    C9                      RET                     ;RETURN WHEN OP1=OP2
                        ;
                                END
```

〈Prog❺〉ブロック・データの左回転

```
                        ;****************************************************
                        ;****          ROTATE  LEFT                      ****
                        ;****************************************************
                        ;       HL     DATA ADDRESS
                        ;       BC     DATA LENGTH
0000    7E              ROL:    LD     A,(HL)          ;LOAD TOP OF DATA
0001    F5                      PUSH   AF              ;SAVE IT WHILE
0002    54                      LD     D,H             ;DE <-- HL
0003    5D                      LD     E,L             ;
0004    23                      INC    HL              ;ADJUST HL
0005    0B                      DEC    BC              ;ADJUST LENGTH
0006    ED B0                   LDIR                   ;SHIFT DATA LEFT
0008    F1                      POP    AF              ;RESTORE TOP OF DATA
0009    12                      LD     (DE),A          ;PUT IT INTO TAIL
000A    C9                      RET                    ;RETURN
                        ;
                                END
```

〈Prog❻〉ブロック・データの右回転

```
                        ;****************************************************
                        ;****          ROTATATE  RIGHT                   ****
                        ;****************************************************
                        ;       HL     DATA ADDRESS
                        ;       BC     DATA LENGTH
0000    0B              ROR:    DEC    BC              ;ADJUST LENGTH
0001    09                      ADD    HL,BC           ;HL=LAST DATA ADDRESS
0002    7E                      LD     A,(HL)          ;LOAD LAST DATA
0003    F5                      PUSH   AF              ;SAVE IT WHILE
0004    54                      LD     D,H             ;DE <-- HL
0005    5D                      LD     E,L             ;
0006    2B                      DEC    HL              ;ADJUST HL
0007    ED B8                   LDDR                   ;SHIFT RIGHT
0009    F1                      POP    AF              ;RESTORE LAST DATA
000A    12                      LD     (DE),A          ;PUT LAST DATA INTO TOP
000B    C9                      RET                    ;RETURN
                        ;
                                END
```

〈Prog❼〉ブロック・データのディジット単位の左回転

```
                        ;****************************************************
                        ;****          ROTATE DIGIT LEFT                 ****
                        ;****************************************************
                        ;       HL     ADDRESS OF DATA TO BE ROTATED
                        ;       C      DATA LENGTH
0000    7E              RLD:    LD     A,(HL)          ;LOAD TOP OF DATA
0001    07                      RLCA                   ;SWAP DIGIT
0002    07                      RLCA                   ;
0003    07                      RLCA                   ;
0004    07                      RLCA                   ;
0005    06 00                   LD     B,0
0007    09                      ADD    HL,BC           ;GET LAST DATA ADDRESS
0008    41                      LD     B,C             ;DATA LENGTH TO REG. B
0009    2B              RLD0:   DEC    HL              ;BACK DATA POINTER
000A    ED 6F                   RLD                    ;ROTATATE DIGIT LEFT
000C    10 FB                   DJNZ   RLD0            ;DONE ? NO,RLD0
000E    C9                      RET                    ;RETURN
                        ;
                                END
```

〈Prog❽〉ブロック・データのディジット単位の右回転

```
                         ;*********************************************************
                         ;****            ROTATE DIGIT RIGHT                  ****
                         ;*********************************************************
                         ;         HL       ADDRESS OF DATA TO BE ROTATED
                         ;         C        DATA LENGTH
0000      E5             RRD:      PUSH     HL                  ;SAVE ADDRESS WHILE
0001      06 00                    LD       B,0                 ;GET LAST DATA ADDRESS
0003      09                       ADD      HL,BC               ;
0004      2B                       DEC      HL                  ;
0005      7E                       LD       A,(HL)              ;LOAD LAST DATA
0006      E1                       POP      HL                  ;RESTORE ADDRESS
0007      41                       LD       B,C                 ;LENGTH TO REG. B
0008      ED 67          RRD0:     RRD                          ;ROTATE DIGIT RIGHT
000A      23                       INC      HL                  ;
000B      10 FB                    DJNZ     RRD0                ;DONE ? NO,RRD0
000D      C9                       RET                          ;
                         ;
                                   END
```

〈Prog❾〉10進加算

```
                         ;*********************************************************
                         ;****            DECIMAL ADD                         ****
                         ;*********************************************************
                         ;         DE       OPERAND 1 ADDRESS
                         ;         HL       OPERAND 2 ADDRESS
                         ;         C        OPERAND  LENGTH
0000      06 00          DADD:     LD       B,0                 ;
0002      09                       ADD      HL,BC               ;
0003      EB                       EX       DE,HL               ;DE POINTS OP2 LAST DATA
0004      09                       ADD      HL,BC               ;
0005      EB                       EX       DE,HL               ;DE POINTS OP1 LAST DATA
0006      41                       LD       B,C                 ;LENGTH TO REG. B
0007      AF                       XOR      A                   ;CLEAR CARRY FLAG
0008      1B             DADD0:    DEC      DE                  ;BACK OP1 POINTER
0009      2B                       DEC      HL                  ;BACK OP2 POINTER
000A      1A                       LD       A,(DE)              ;LOAD OP1
000B      8E                       ADC      A,(HL)              ;ADD OP1 AND OP2
000C      27                       DAA                          ;DECIMAL ADJUSTING
000D      12                       LD       (DE),A              ;STORE RESULT INTO OP1
000E      10 F8                    DJNZ     DADD0               ;DONE ? NO,DADD0
0010      C9                       RET                          ;
                         ;
                                   END
```

〈Prog❿〉10進減算

```
                         ;*********************************************************
                         ;****            DECIMAL SUBTRACT                    ****
                         ;*********************************************************
                         ;         DE       OPERAND 1 ADDRESS
                         ;         HL       OPERAND 2 ADDRESS
                         ;         C        OPERAND  LENGTH
0000      06 00          DSUB:     LD       B,0                 ;
0002      09                       ADD      HL,BC               ;
0003      EB                       EX       DE,HL               ;DE POINTS OP2 LAST DATA
0004      09                       ADD      HL,BC               ;
0005      EB                       EX       DE,HL               ;DE POINTS OP1 LAST DATA
0006      41                       LD       B,C                 ;LENGTH TO REG. B
0007      AF                       XOR      A                   ;CLEAR CARRY FLAG
0008      1B             DSUB0:    DEC      DE                  ;BACK OP1 POINTER
0009      2B                       DEC      HL                  ;BACK OP2 POINTER
```

```
000A    1A              LD      A,(DE)          ;LOAD OP1
000B    9E              SBC     A,(HL)          ;SUBTRACT OP2 FROM OP1
000C    27              DAA                     ;DECIMAL ADJUSTING
000D    12              LD      (DE),A          ;STORE RESULT INTO OP1
000E    10 F8           DJNZ    DSUBO           ;DONE ? NO,DSUBO
0010    C9              RET                     ;
                ;
                        END
```

〈Prog⓫〉テーブル・ジャンプ

```
                ;**************************************************
                ;****           TABLE JUMP                    ****
                ;**************************************************
                ;       L       VECTOR VALUE
0000    26 00   JUMP:   LD      H,0             ;ZERO TO H
0002    29              ADD     HL,HL           ;DOUBLE VECTOR VALUE
0003    EB              EX      DE,HL           ;2 X VECTOR TO DE
0004    21 000D         LD      HL,TABLE        ;LOAD TABLE ADDRESS
0007    19              ADD     HL,DE           ;LOAD ENTRY ADDRESS
0008    5E              LD      E,(HL)          ;LOAD LOWER ADDRESS
0009    23              INC     HL              ;BUMP TABLE POINTER
000A    56              LD      D,(HL)          ;LOAD UPPER ADDRESS
000B    EB              EX      DE,HL           ;ROUTINE ADDRESS TO HL
000C    E9              JP      (HL)            ;JUMP TO EACH ROUTINE
                ;
000D    0D02    TABLE:  DW      ROUT00          ;ROUTINE 00 (C=00)
000F    1D02            DW      ROUT01          ;ROUTINE 01 (C=01)
0011    2D02            DW      ROUT02          ;ROUTTNE 02 (C=02)
                ;
                ;
                        ORG     TABLE+253*2
0207    2D12            DW      ROUTFD          ;ROUTINE FD (C=FD)
0209    3D12            DW      ROUTFE          ;ROUTINE FE (C=FE)
020B    4D12            DW      ROUTFF          ;ROUTINE FF (C=FF)
                ;
020D            ROUT00:                         ;ROUTINE 00
                        ORG     $+10H
021D            ROUT01:                         ;ROUTINE 01
                        ORG     $+10H
022D            ROUT02:                         ;ROUTINE 02
                        ORG     $+4096
122D            ROUTFD:                         ;ROUTINE FD
                        ORG     $+10H
123D            ROUTFE:                         ;ROUTINE FE
                        ORG     $+10H
124D            ROUTFF:                         ;ROUTINE FF
                ;
                        END
```

〈Prog⓬〉コード変換

```
                ;**************************************************
                ;****           TRANSLATE                     ****
                ;**************************************************
                ;       C       CODE TO BE TRANSLATED
                ;       A       CODE TRANSLATED
0000    06 00   XLAT:   LD      B,0             ;ZERO TO REG. B
0002    21 0800         LD      HL,TABLE        ;LOAD TRANSLATION TABLE
                                                ;ADDRESS
0005    09              ADD     HL,BC           ;GET TABLE ENTRY ADDRESS
0006    7E              LD      A,(HL)          ;LOAD TRANSLATED CODE
0007    C9              RET                     ;RETURN
                ;
0008            TABLE:  DS      256             ;TRANSLATION TABLE
                ;
                        END
```

〈Prog⓭〉 コード・デコーダ

```
                              ;**************************************************
                              ;****         CODE DECODER                       ****
                              ;**************************************************
                              ;        A        CODE TO BE DECODED
0000      DD 21 1C00          CDEC:    LD       IX,TABLE          ;LOAD TABLE ADDRESS
0004      01 0300                      LD       BC,3              ;LOAD TABLE WIDTH
0007      DD CB 00 7E         CDEC0:   BIT      7,(IX+0)          ;TABLE END ?
000B      C0                           RET      NZ                ;NOT FOUND RETURN
000C      DD BE 00                     CP       (IX+0)            ;CODE MATCH ?
000F      28 04                        JR       Z,CDEC1           ;YES,CDEC1
0011      DD 09                        ADD      IX,BC             ;IX POINTS NEXT ENTRY
0013      18 F2                        JR       CDEC0             ;
0015      DD 6E 01            CDEC1:   LD       L,(IX+1)          ;LOAD LOWER ADDRESS
0018      DD 66 02                     LD       H,(IX+2)          ;LOAD UPPER ADDRESS
001B      E9                           JP       (HL)              ;GO TO EACH ROUTINE
                              ;
001C      41                  TABLE:   DB       'A'               ;
001D      2F00                         DW       ROUTA             ;A ROUTINE
001F      42                           DB       'B'               ;
0020      3F00                         DW       ROUTB             ;B ROUTINE
0022      50                           DB       'P'               ;
0023      3F10                         DW       ROUTP             ;P ROUTINE
0025      53                           DB       'S'               ;
0026      4F10                         DW       ROUTS             ;S ROUTINE
0028      59                           DB       'Y'               ;
0029      4F20                         DW       ROUTY             ;Y ROUTINE
002B      5A                           DB       'Z'               ;
002C      5F20                         DW       ROUTZ             ;Z ROUTINE
                              ;
002E      80                           DB       080               ;TABLE SENTINEL
                              ;
002F                          ROUTA:                              ;A ROUTINE
                                       ORG      $+10H
003F                          ROUTB:                              ;B ROUTINE
                                       ORG      $+1000H
103F                          ROUTP:                              ;P ROUTINE
                                       ORG      $+10H
104F                          ROUTS:                              ;S ROUTINE
                                       ORG      $+1000H
204F                          ROUTY:                              ;Y ROUTINE
                                       ORG      $+10H
205F                          ROUTZ:                              ;Z ROUTINE
                              ;
                                       END
```

〈Prog⓮〉 ASCII→16進変換

```
                              ;**************************************************
                              ;****         ASCII TO HEX CONVERTION            ****
                              ;**************************************************
                              ;        A        ASCCII CHARACTER        (0-9,A-F)
                              ;        HEX VALUE WILL BE RETURNED IN LOWER 4 BITS OF A.
0000      FE 30               ATOH:    CP       '0'               ;LOWER THAN '0' ?
0002      D8                           RET      C                 ;YES,ERROR RETURN
0003      FE 3A                        CP       '9'+1             ;0-9 ?
0005      38 09                        JR       C,ATOH0           ;YES,ATOH0
0007      FE 41                        CP       'A'               ;LOWER THAN 'A' ?
0009      D8                           RET      C                 ;YES,ERROR RETURN
000A      FE 47                        CP       'F'+1             ;A-F ?
000C      3F                           CCF                        ;COMPLEMENT CARRY
000D      D8                           RET      C                 ;NO,ERROR RETURN
000E      C6 09                        ADD      A,09H             ;ADJUST A-F
```

```
0010    E6 0F        ATOHO:  AND     0FH             ;CLEAR UPPER 4 BITS
0012    C9                   RET                     ;
                     ;
                             END
```

〈Prog⑮〉 16進→ASCII変換

```
                     ;***********************************************************
                     ;****           HEX TO ASCII CONVERTION                  ****
                     ;***********************************************************
                     ;       A       HEXADECIMAL VALUE
                     ;       BC      ASCII CHARACTER (RETURNED)
0000    F5           HTOA:   PUSH    AF              ;SAVE HEX VALUE WHILE
0001    07                   RLCA                    ;SWAP DIGIT
0002    07                   RLCA                    ;
0003    07                   RLCA                    ;
0004    07                   RLCA                    ;
0005    CD 0011              CALL    GASC            ;GET ASCII CHARACTER
0008    47                   LD      B,A             ;ASCII CHARACTER TO B
0009    F1                   POP     AF              ;RESTORE HEX VALUE
000A    E6 0F                AND     0FH             ;CLEAR UPPER 4 BITS
000C    CD 0011              CALL    GASC            ;GET ASCII CHARACTER
000F    4F                   LD      C,A             ;ASCII CHARACTER TO C
0010    C9                   RET                     ;RETURN
                     ;
                     ;       GET ASCII CHARACTER SUBROUTINE
                     ;       A       HEX DIGIT IN LOWER 4 BITS
                     ;       ASCCII CHARACTER WILL BE RETURNED IN REG. A
0011    FE 0A        GASC:   CP      0AH             ;00-09 ?
0013    38 02                JR      C,GASC0         ;YES,GASC0
0015    C6 07                ADD     A,07H           ;ADJUST 0A-0F
0017    C6 30        GASC0:  ADD     A,30H           ;GET ASCII CHARACTER
0019    C9                   RET                     ;RETURN
                     ;
                             END
```

〈Prog⑯〉 ストリング・データの表示

```
                     ;***********************************************************
                     ;****           CONSOLE OUT STRING DATA (1)              ****
                     ;***********************************************************
0000    21 0600              LD      HL,MSG          ;LOAD MESSAGE ADDRESS
0003    CD 1900              CALL    DISP            ;DISPLAY MESSAGE
                     ;
0006    54 48 49 53  MSG:    DB      'THIS IS AN MESSAGE',0FFH
000A    20 49 53 20
000E    41 4E 20 4D
0012    45 53 53 41
0016    47 45 FF
                     ;
0019    7E           DISP:   LD      A,(HL)          ;LOAD MSG DATA
001A    FE FF                CP      0FFH            ;END OF MESSAGE ?
001C    C8                   RET     Z               ;YES,RETURN
001D    CD 0000              CALL    COUT            ;DATA TO CONSOLE
0020    23                   INC     HL              ;BUMP MSG POINTER
0021    18 F6                JR      DISP            ;
                     ;
                     ;***********************************************************
                     ;****           CONSOLE OUT STRING DATA (2)              ****
                     ;***********************************************************
0023    CD 3900              CALL    DISPX           ;DISPLAY MESSAGE
0026    54 48 49 53          DB      'THIS IS AN MESSAGE',0FFH
002A    20 49 53 20
002E    41 4E 20 4D
```

```
0032    45 53 53 41
0036    47 45 FF
                      ;
0039    E3            DISPX:   EX     (SP),HL      ;HL POINTS MESSAGE
003A    7E            DISPXO:  LD     A,(HL)       ;LOAD MESSAGE DATA
003B    FE FF                  CP     OFFH         ;END OF MESSAGE ?
003D    23                     INC    HL           ;BUMP MESSAGE POINTER
003E    28 05                  JR     Z,DISPX1     ;YES,DISPX1
0040    CD 0000                CALL   COUT         ;DATA TO CONSOLE
0043    18 F5                  JR     DISPXO       ;
0045    E3            DISPX1:  EX     (SP),HL      ;RETURN ADDRESS TO STACK
0046    C9                     RET                 ;RETURN
                      ;
                               END
```

第3章

Z80ファミリLSI編

3.0 Z80ファミリの周辺LSI

▶Z80周辺LSIの特徴◀

Z80ファミリの周辺LSI（ペリフェラル）として発表され，かつ現在入手可能なものはZ80 PIO，Z80 CTC，Z80 SIO，Z80 DMA，それにZ80 DARTがあります．図1にZ80ファミリの周辺LSIの一覧表を示します．

Z80ファミリとは，日本語でいえば「Z80一家」ということになり，これらのLSIはZ80 CPU親分を中心とした子分であり，かつZ80一家を構成する大事な子分達であるということができます．

Z80ファミリの周辺LSIの名称はZ80 CPUと同じように，動作可能なクロック速度に応じてZ80，Z80A，またはZ80Bというヘッドが付きます．Z80のあとにAもBも付かないものを"ノンAバージョン(Non A version)"，Aが付くものをAバージョン，Bが付くものをBバージョンと呼びます．

当初はノンAバージョンとAバージョンしかなかったので，ノンAバージョンという呼びかたも正しかったのですが，Bバージョンがある現在，"ノンA／Bバージョン"と呼んだほうが適切なのかもしれません．ノンAバージョンの最高クロック速度は2.5MHz，Aバージョンは4 MHz，Bバージョンは6 MHzです．

Z80周辺LSIの特徴

Z80周辺LSIを他のファミリ，例えばインテル系の8080，8085ファミリの周辺LSIと比較してみて気が付くもっとも大きな特徴は，デイジー・チェイン式の優先ベクトル割り込み方式でしょう．

ご存知のようにZ80 CPUにはモード0～2という3タイプの割り込み形式があります．モード0およびモード1は，Z80 CPUが発表された当時，一世を風靡していた8080とのコンパチビリティを持たせるために備えられた割り込みモードであり，Z80 CPUとしては不本意な割り込みモードです．とはいえ揺籃期のZ80 CPUにとっては不可欠のモードあることは確かでした．

Z80 CPUにとって本命の割り込みモードは，モード2ベクトル割り込み方式です．このモード2ベクトル割り込み方式では，CPUの割り込み認識シーケンス（インタラプト・アクノウリッジ・シーケンス）において，割り込みをかけた周辺LSIが8ビットの割り込みベクトルをCPUに対して送ります．

CPUはこの割り込みベクトルの値と内部のインタラプト・レジスタ（IR）とにより，メモリ上の割り込みルーチン・アドレス・テーブルをアクセスし，このテーブルに記述されたアドレスの割り込みルーチンにプログラムの制御を移します．

したがってZ80ファミリの周辺LSIは，すべてこのモード2割り込みを行う機能を持っています．図2にZ80 CPUのモード2割り込みの方法を示します．

またZ80ファミリの周辺LSI間の割り込み優先順位制御方式は，Z80ファミリ固有のものであり，大きな特徴として位置づけられるものです．Z80ファミリの周辺LSI間の割り込み優先順位制御は，割り込みデイジー・チェインにより行われます．図3にZ80ファ

〈図1〉
Z80ファミリの周辺LSI
（○：入手可
×：入手不可）

LSI名	フルネーム	Z80A	Z80B	機能
Z80 PIO	Parallel Input Output	○	○	パラレルI/Oインターフェース
Z80 CTC	Counter Timer Circuit	○	○	カウンタ/タイマ
Z80 SIO	Serial Input Output	○	○	シリアルI/Oインターフェース
Z80 DMA	Direct Memory Access	○	×	ダイレクト・メモリ・アクセス・コントローラ
Z80 DART	Dual Asynchronous Receiver/Transmitter	○	○	2チャネル 非同期レシーバ/トランスミッタ

ミリの周辺LSIを結ぶ割り込みデイジー・チェインの例を示します.

Z80ファミリの割り込みデイジー・チェインは各周辺LSIのIEO(インタラプト・イネーブル・アウト)出力を,そのLSIより割り込み優先順位の低いLSIのIEI(インタラプト・イネーブル・イン)入力に接続します.そして最高の割り込み優先順位を持つLSIのIEI入力は常に"H"とします.

各LSIは自分のIEI入力の状態が"H"のときに限り,CPUに対して割り込み要求を出すことができ,割り込み要求を出すとIEO出力を"L"にし,自分よりも割り込み優先順位の低いLSIの割り込み要求を抑えます.

CPUに割り込み要求を許可されたLSI(IEI="H",$\overline{IORQ}$="L",$\overline{M1}$="L")は,割り込みルーチン(プログラム)がRETI命令を実行するのを監視しており,同命令が実行されるとIEO出力を"H"にして自分より優先度の低いLSIの割り込み要求を許可します.

このようにCPUが実行する命令を盗み見て割り込み制御を行う点もZ80ファミリ周辺LSIの特徴です.

以上のデイジー・チェイン方式の優先ベクトル割り込み機能を,Z80ファミリの周辺LSIはすべて持っています.したがってZ80ファミリにはインタラプト・コントローラというLSIは存在しません.

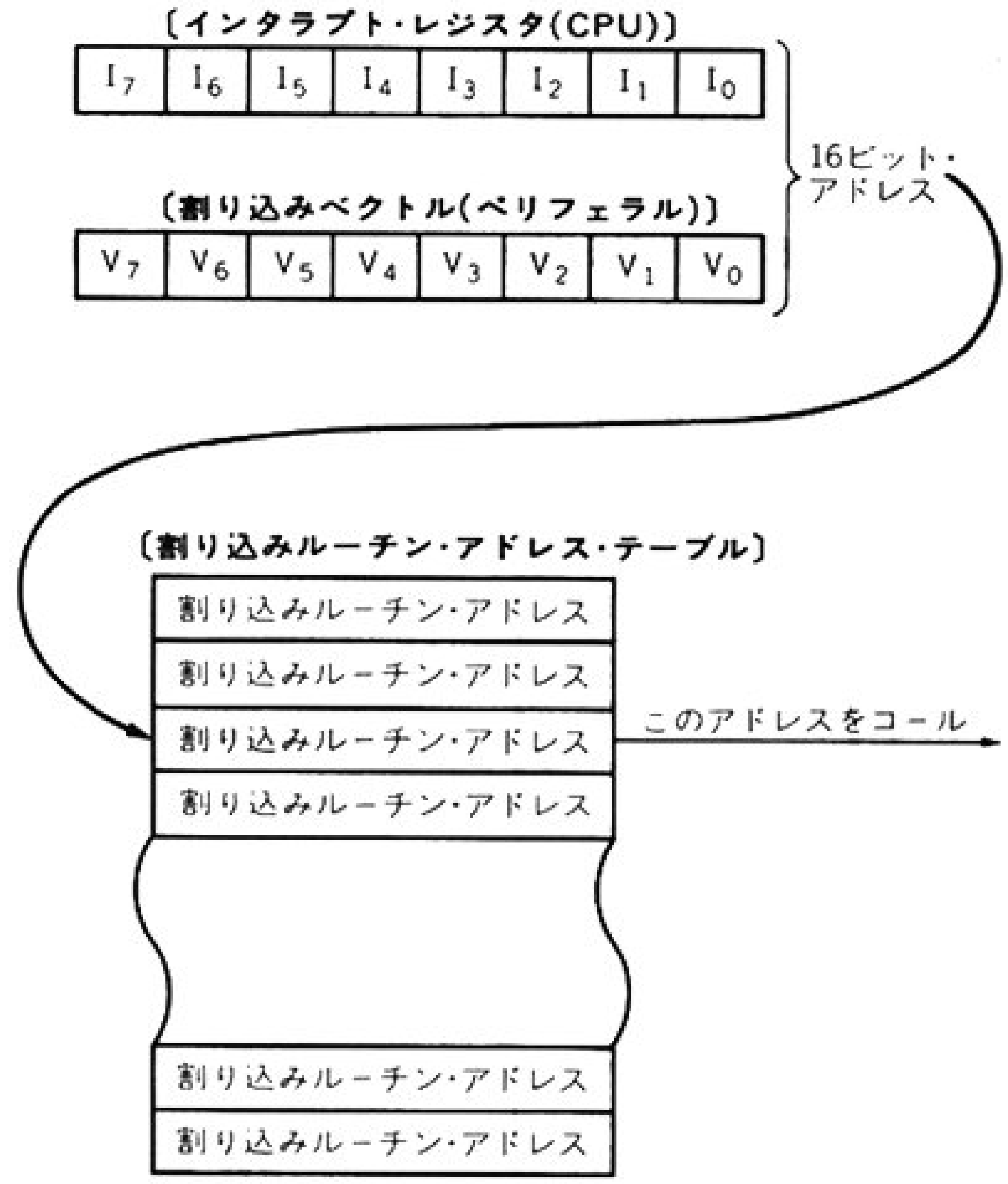
割り込み以外のZ80周辺LSIが持つ特徴はブロック・トランスファ命令(OTIR)により,プログラミングができることです.SIO, DMA, DARTなどは機能が複雑であるため,コントロール・レジスタの数が多いのですが,コントロール・レジスタに対して割り当てられているI/Oポートはわずか1個にすぎません.

ではどのようにしてコントロール・レジスタの指定を行うのかというと,コマンドの中に次のデータが対象とするレジスタ番号を示す値(ポインタ)をセットします.

これにより一連のコマンド列(コマンド・チェイン)をデータとして定義しておき,一発のOTIR命令を出せば一連のコマンド列がLSIに対して送られ(同一ポートに対して),コマンド列中にあるポインタにしたがって次つぎと必要なコントロール・レジスタに対してコマンドが書き込まれることになります.

図4にZ80周辺LSIに対するコマンド・チェインの送出方法を示します.

〈図2〉 Z80 CPUのモード2割り込み

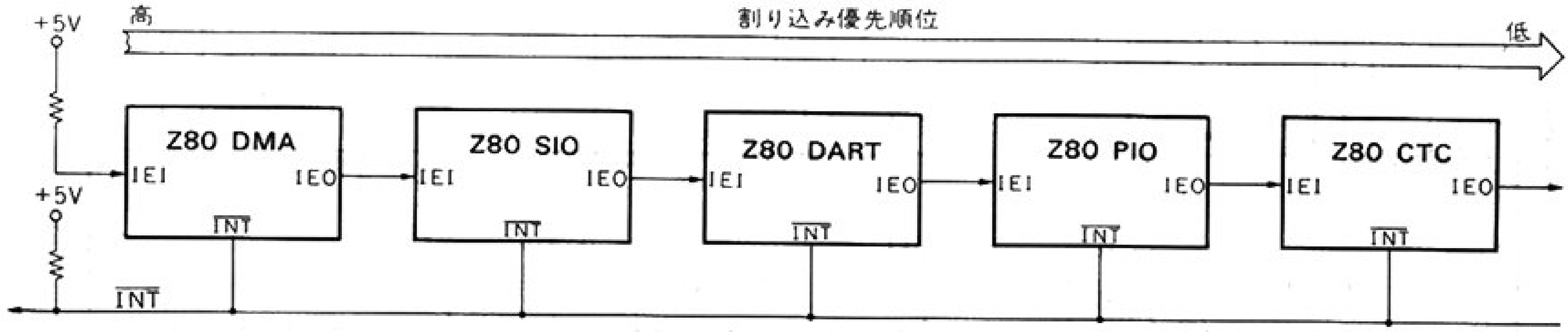


Z80ファミリのLSIとZ80 CPUとのインターフェース

Z80ファミリのLSIとZ80 CPU間のインターフェースはDMAを除くとみな同じです.図5にZ80ファミリのLSIとZ80 CPU間のインターフェース例を示します.

同じファミリにありながら,DMAだけなぜ違うのかというと,DMAはその機能上CPUと同じ立場にあるからです.したがって図5のようにCPUと並べて書くとどちらがCPUでどちらがDMAであるのかほとんど見分けがつかない程度に似ています.

〈図3〉 Z80ファミリの割り込みデイジー・チェイン



Z80 ファミリ LSI のあらまし

■Z80 PIO

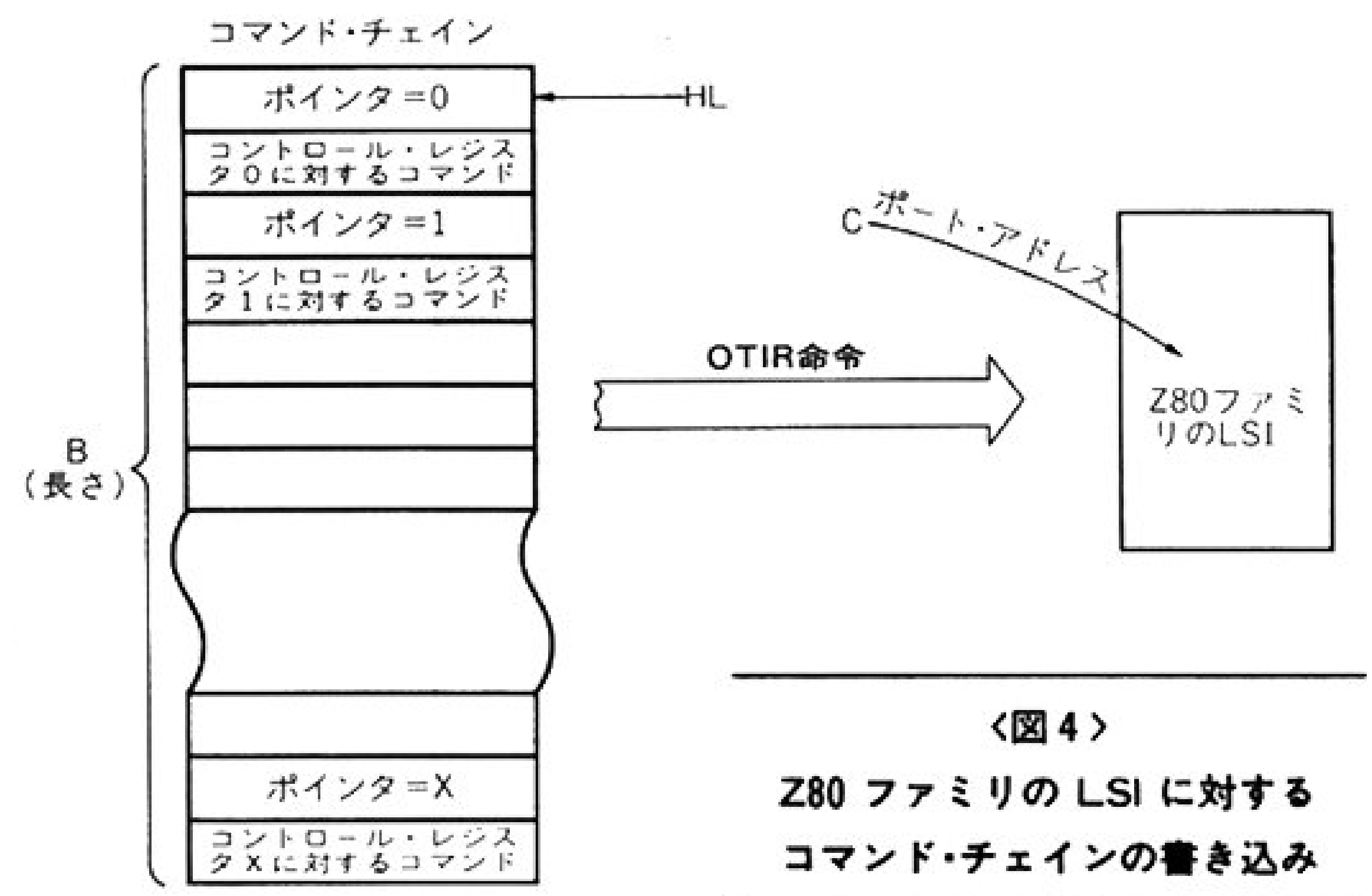
Z80 PIO は2ポート構成（ポートA／ポートB）のパラレル・インターフェースであり，モード0～モード3の4種類のモードを持っています．それぞれは，

・モード0：出力モード
・モード1：入力モード
・モード2：入／出力モード
・モード3：ビット・モード

で各ポートはモード2を除いて独立にモード設定可能ですが，モード2に設定できるのはポートAだけです．

ポートAをモード2にするときには，ポートBはモード3でしか使えません．また，モード3以外のモードでは，レディおよびストローブによりハンド・シェークができます．

図6に Z80 PIO の構成およびピン接続図を示します．



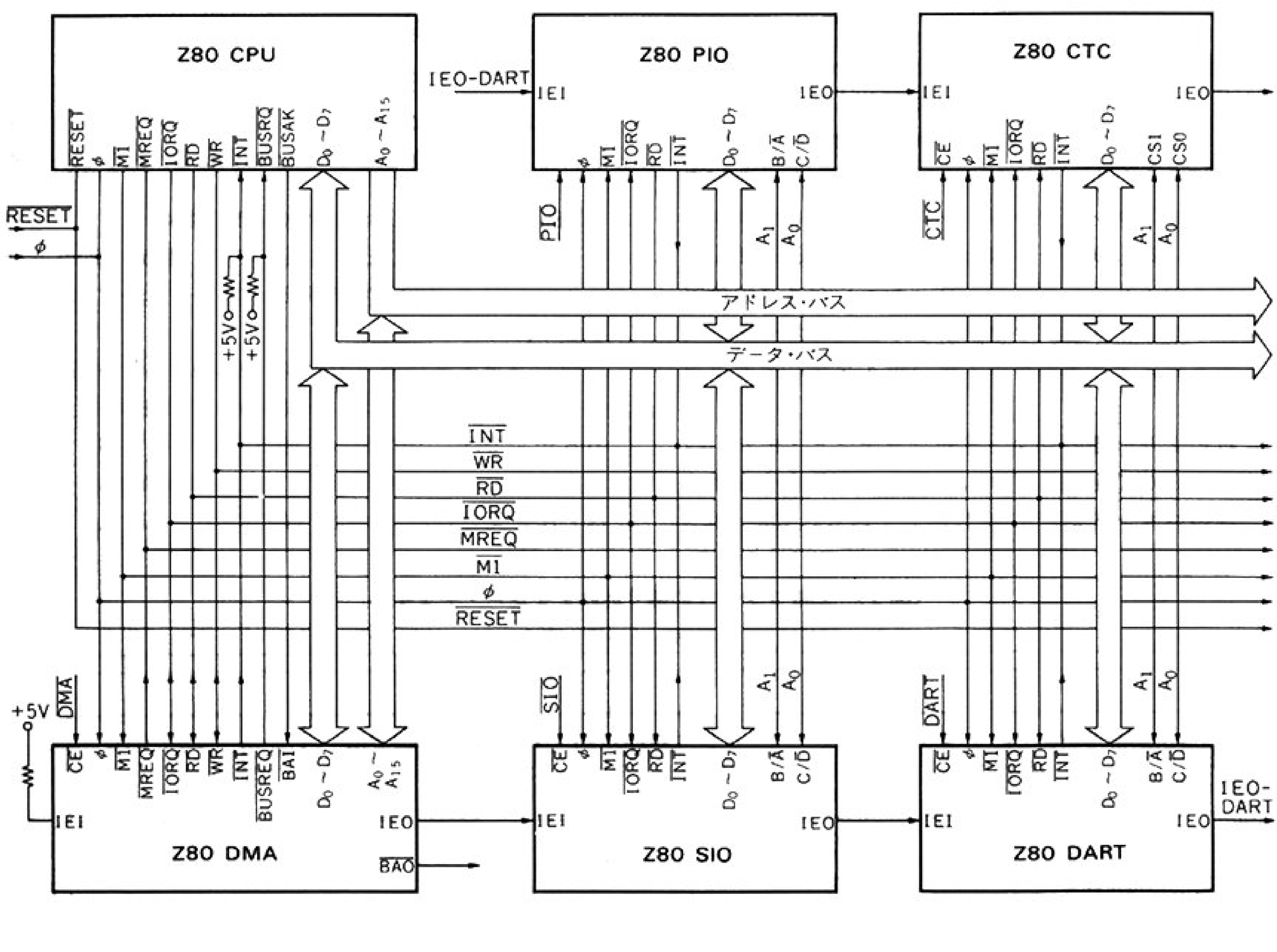
〈図4〉 Z80 ファミリの LSI に対するコマンド・チェインの書き込み

■Z80 CTC

Z80 CTC は4チャネルのカウンタ/タイマです．各チャネルはシステム・クロック ϕ を16分の1または256分の1に分周できるプリスケーラと，8ビットのダウン・カウンタを持っています．

動作モードとしては，タイマ・モードとカウンタ・モードがあり，タイマ・モードではプリスケーラの出力により8ビット・カウンタをカウント・ダウンしま

〈図5〉 Z80 ファミリ LSI と Z80 CPU のインターフェース



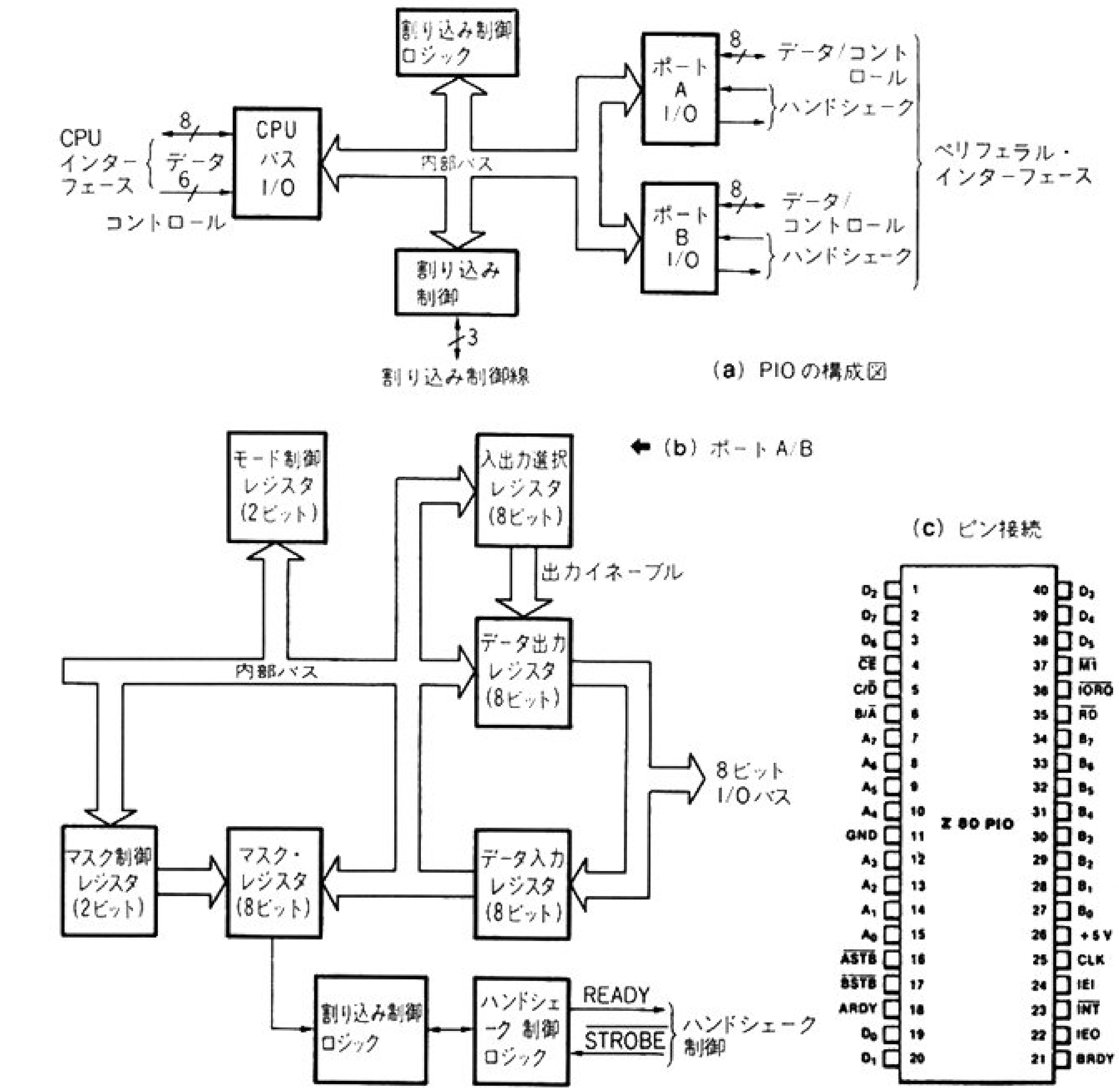
す．カウンタ・モードでは各チャネルの外部クロック入力を8ビット・カウンタにより，カウント・ダウンします．

8ビット・カウンタがゼロになると，チャネルのカウンタ出力にパルスが出力され，このとき同時にCPUに対して割り込みを発生することができます．Z80 CTCはシステム・タイマやボーレート・ジェネレータなどに用いることができます．

図7にZ80 CTCの構成，およびピン接続図を示します．

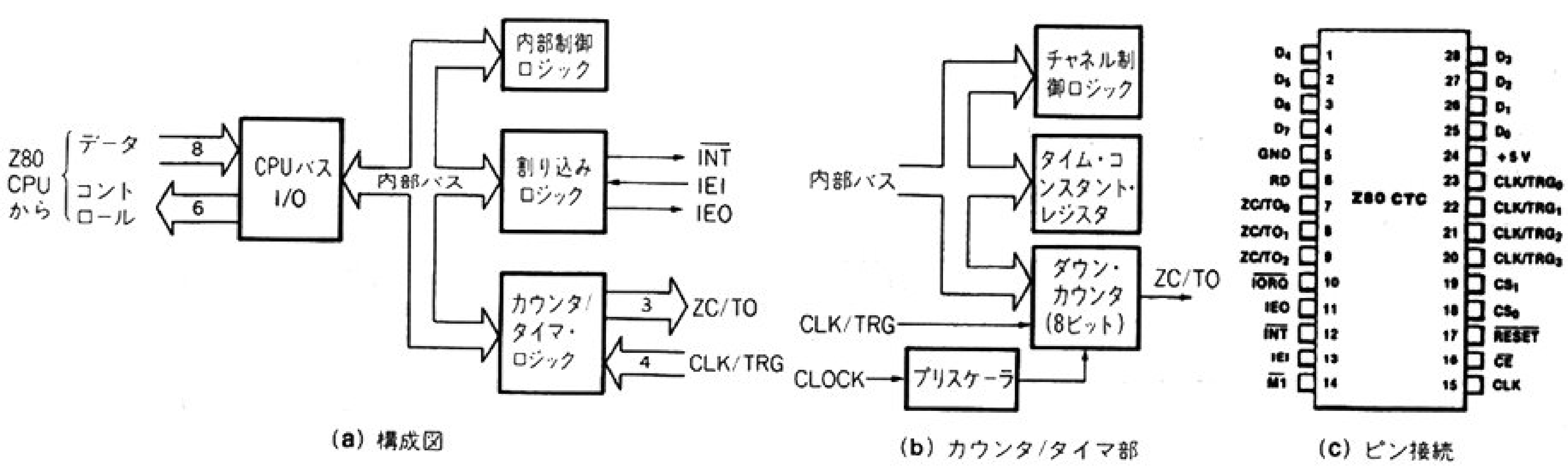
■Z80 SIO

Z80 SIOはまったく独立に動作できる全2重チャネルを二つ持ったシリアル・インターフェースです．このSIOは非同期モード，同期モードはもちろんのこと，SDLCの制御までも可能な，マルチプロトコル・コミ



〈図6〉
Z80 PIOの構成図と
ピン接続図*

〈図7〉 Z80 CTCの構成図とピン接続図



ユニケーション・コントローラということができます．

このようにSIOはいろいろなことができるため，コントロール・レジスタが複雑であり，使いこなすのに骨がおれますが，じっくり腰を落ち着けてがんばれば，誰にでも使えるLSIであることが，お分りいただけると思います．

図8にZ80 SIOの構成図を示します．

■Z80 DMA

Z80 DMAは1チャネルのDMAコントローラですが，ソース・アドレス，およびデスティネーション・アドレスがプログラミングできますので，同時でなければいくつかのデータ転送パスを制御できます．

このZ80 DMAはふつうのメモリ ↔ I/O のデータ転送のほかに，メモリ ↔ メモリ，I/O ↔ I/O のデータ転送も可能であり，かつサーチ動作もできます．Z80 DMAの転送速度の最大値は2Mバイト／秒です（4MHz，2サイクル・タイミングを用いたとき）．

図9にZ80 DMAの構成，およびピン接続図を示します．

■Z80 DART

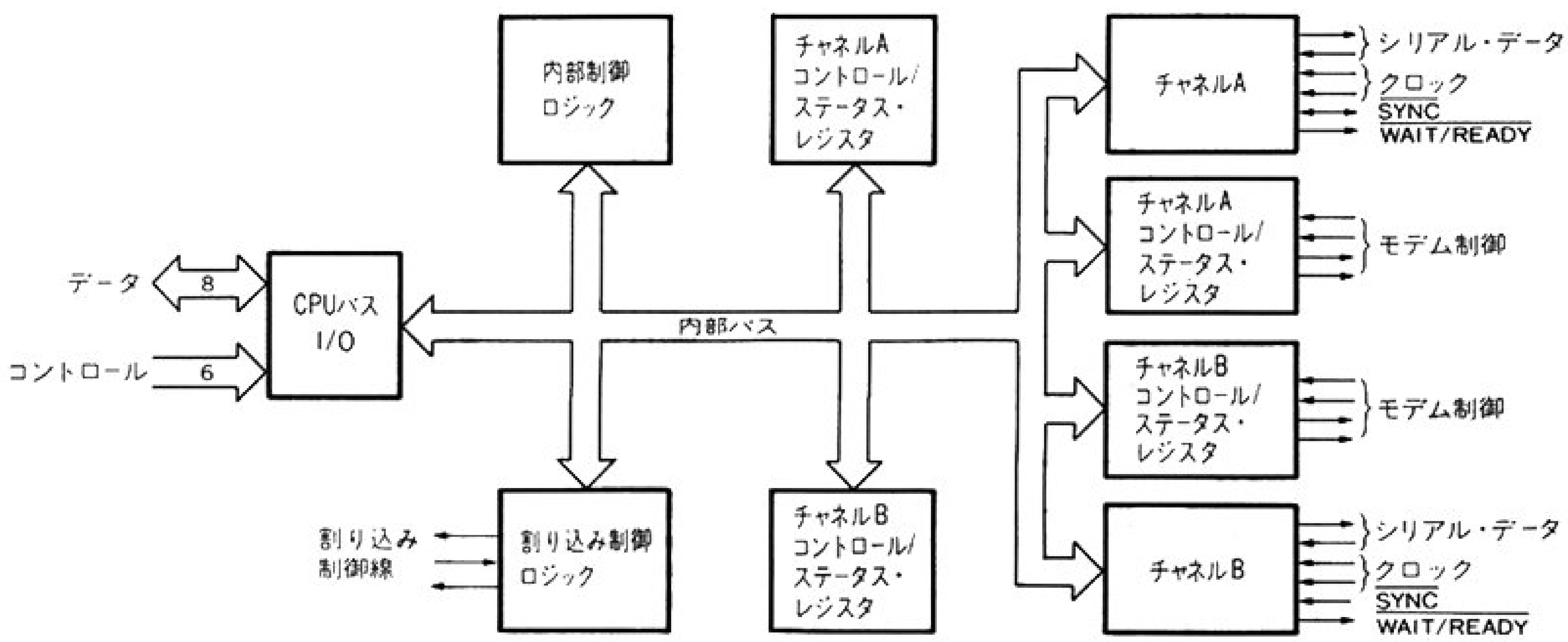
Z80 DARTはZ80ファミリの中で一番最後に発表になったLSIであり，Dual Asynchronous Receiver/Transmitterです．つまり非同期通信チャネルを二つ持ったLSIです．このZ80 DARTはZ80 SIOの機能のうちの，非同期モードの部分だけを持ったLSIです．

したがってピン配置およびコントロール・レジスタの構成はZ80 SIOとまったく同じです．ただしこのDARTは非同期モード専用ですので，SIOの $\overline{\text{SYNCA}}$/$\overline{\text{SYNCB}}$ の機能をリング・インジケータの検出に用いています．

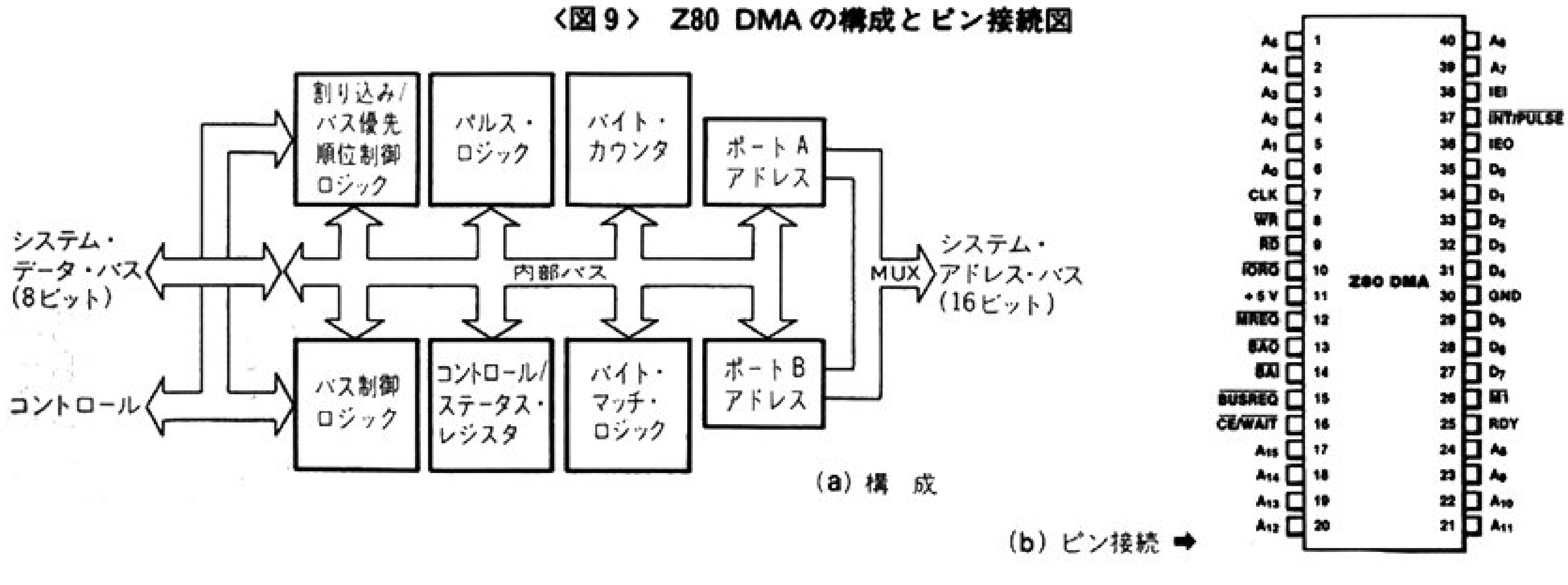
Z80 DARTが発売された背景には，Z80ファミリにはシリアル・インターフェースはSIOだけしかないので，非同期通信しかやらないユーザーにはSIOは高くつきすぎ，このようなユーザーは仕方なく8251Aを使っていたというようなことが考えられます．

図10にZ80 DARTのピン接続図を示します（構成図は図8参照）．

〈図8〉 Z80 SIOの構成図(Z80 DARTも同じ)



〈図9〉 Z80 DMAの構成とピン接続図



(a) 構 成

(b) ピン接続 ➡

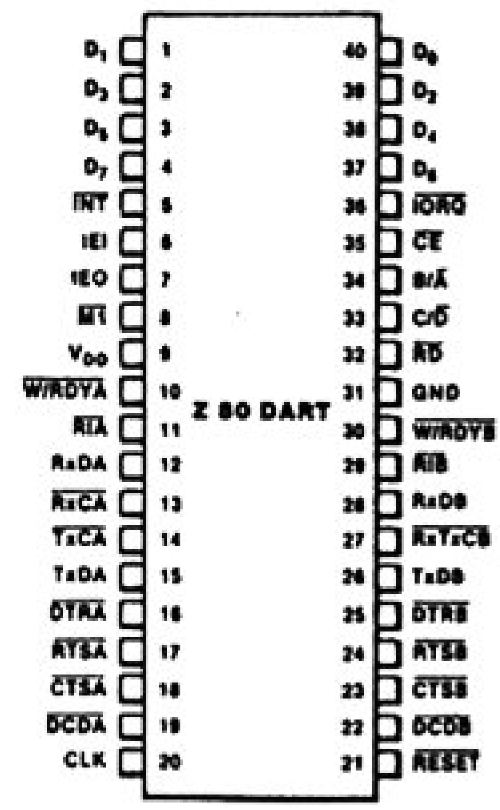
以上5種類の周辺LSIのほかに，ザイログ社ではユニバーサル・ペリフェラルと称して，SCC(Serial Communications Controller)，ASCC(Asynchronous Serial Communications Controller)，CIO(Counter/Timer and Parallel I/O Unit)という周辺LSIを提供しています．

SCCおよびASCCはZ80ファミリのSIO，およびDARTに相当し，CIOはCTCとPIOを合体させたようなものです．これらのLSIの持つ機能は，基本的にはZ80ファミリのそれと同じですが，開発された時期が比較的最近なので，内容は非常に洗練されたものになっています．

これらのLSIはCPUクロックを用いず，独立した非同期のクロックで動作するように設計されており，CPUとのデータのやりとりを制御する信号は$\overline{\mathrm{RD}}$と$\overline{\mathrm{WR}}$だけです．したがって，これらのLSIは基本的にはどのようなCPUともインターフェースをとることができます．

またこれらのLSIもZ80ファミリと同じく，デイジー・チェイン式の優先割り込み制御構造をとっていますが，おのおの$\overline{\mathrm{INTACK}}$(インタラプト・アクノウリッジ)入力を持っており，かつプログラムによりデイジー・チェインの制御(Z80CPUのRETI機能)をすることができますので，どのようなCPUを用いても，あのすばらしいデイジー・チェインの恩恵にあずかることができます．

〈図10〉
Z80 DARTのピン接続



3.1 Z80 PIO編

▶Z80 Parallel I/O◀

Z80 PIO(Parallel I/O)は，Z80ファミリの周辺LSIの中で最初に発表されたものであり，その機能はi8255およびMC6821とほとんど同じです．どちらかといえばMC6821によく似ています．Z80 PIOの概要を箇条書にして記すと次のようになります．

- 5V単一電源である．
- 40ピンDIPである．
- 2ポート構成である．
- 四つのモードを持っている．
  - (1)出力モード
  - (2)入力モード
  - (3)双方向モード
  - (4)ビット・モード
- 各ポートにハンドシェーク機能がある．
- Z80 CPUのモード2割り込みベクトルの発生機能を内蔵している．
- デイジー・チェイン式割り込み制御ロジックを内蔵している．
- 1ポート(B)はダーリントン・トランジスタをドライブできる．
- すべての入・出力ラインはTTLコンパチブルである．

Z80 PIOの構成

図1にPIO全体の構成図を示します．また図2にポートA，およびポートBの詳細を示します．

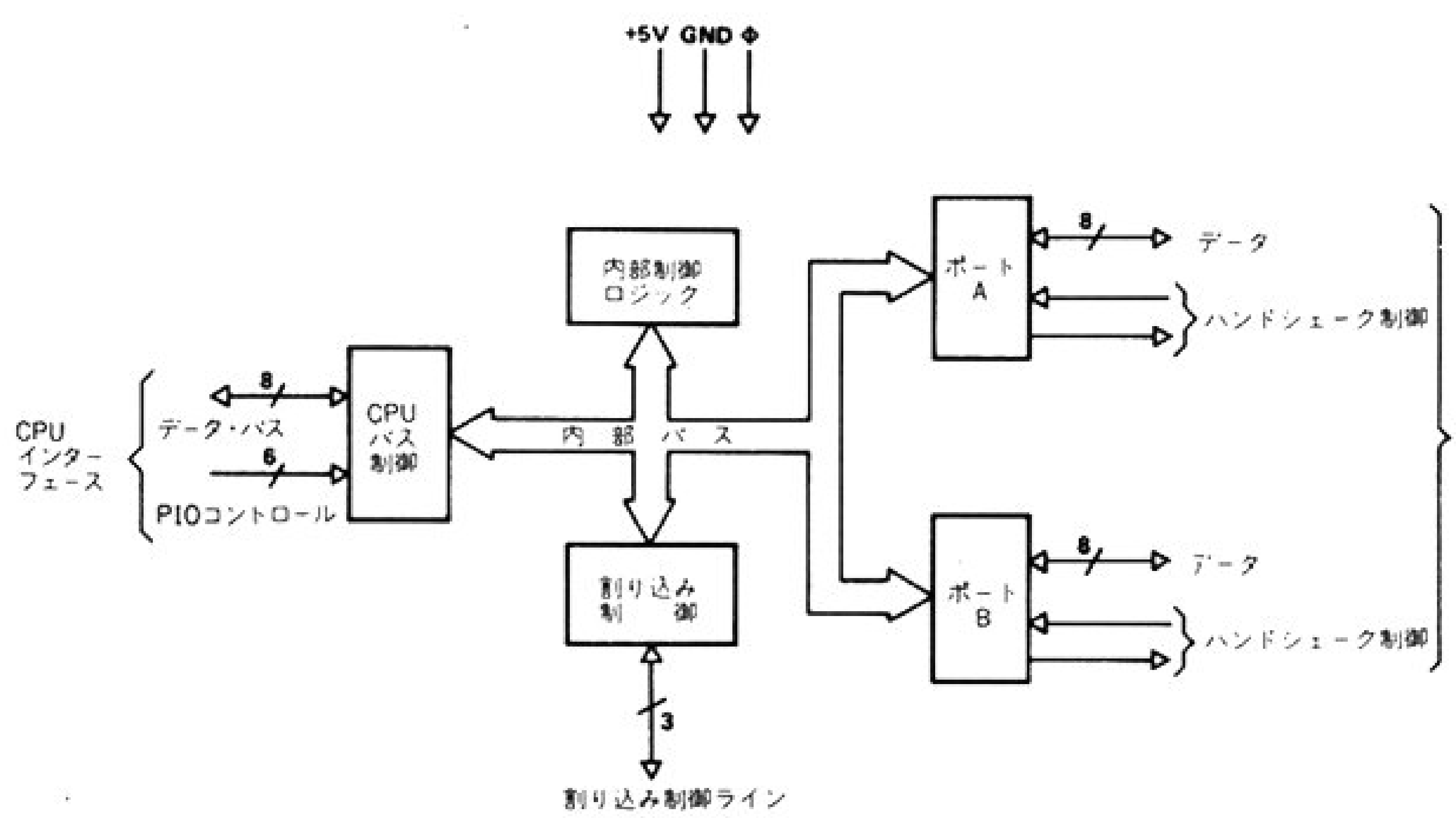
CPUバス制御部はZ80 CPUとのコミュニケーションを管理し，内部制御ロジックはCPUバス制御部と共同で割り込み制御部，およびA/B両ポートを制御します．割り込み制御部は，割り込みデイジー・チェインおよび割り込みベクトルの発生を制御するロジックです．

各ポートは，7個のレジスタとハンドシェーク制御ロジックとからなります．これらのレジスタは，

(1) データ入力レジスタ
(2) データ出力レジスタ
(3) モード制御レジスタ
(4) マスク・レジスタ
(5) 入・出力選択レジスタ
(6) マスク制御レジスタ
(7) 割り込みベクトル・レジスタ

の七つです．

〈図1〉
Z80 PIOのブロック・ダイヤグラム
(ポートA，およびポートBの詳細については図2を参照のこと)



モード制御レジスタは2ビットで，前述の四つの動作モードのうちの一つを選択するのに用いられます．CPUとポートの先に接続された周辺装置との間のデータ転送は，すべてデータ出力レジスタとデータ入力レジスタを介して行われます．データ出力レジスタに書き込まれたデータをCPUは読むことができます．

したがって，データ出力レジスタにデータを書き込み，その後その内容を読むことによりPIOの機能の一部をテストすることができます．また，各ポートのハンドシェーク制御ラインはPIOと周辺装置間のデータのやりとりを制御します．

マスク・レジスタ，マスク制御レジスタ，および入・出力選択レジスタは，ビット・モードにおいてのみ用いられます．このモードでは各ポートの8本のデータ・ラインを個々に入力，または出力ラインとして指定できます．この指定を行うのが入・出力選択レジスタです．マスク・レジスタはビット・モードの割り込みに関連して用いられます．

ビット・モードでは，マスクのかかっていない任意のピンの状態が指定した状態（“H” または “L”）になったときに，割り込みを発生させることができます．この割り込みマスクを保持するのがマスク・レジスタです．

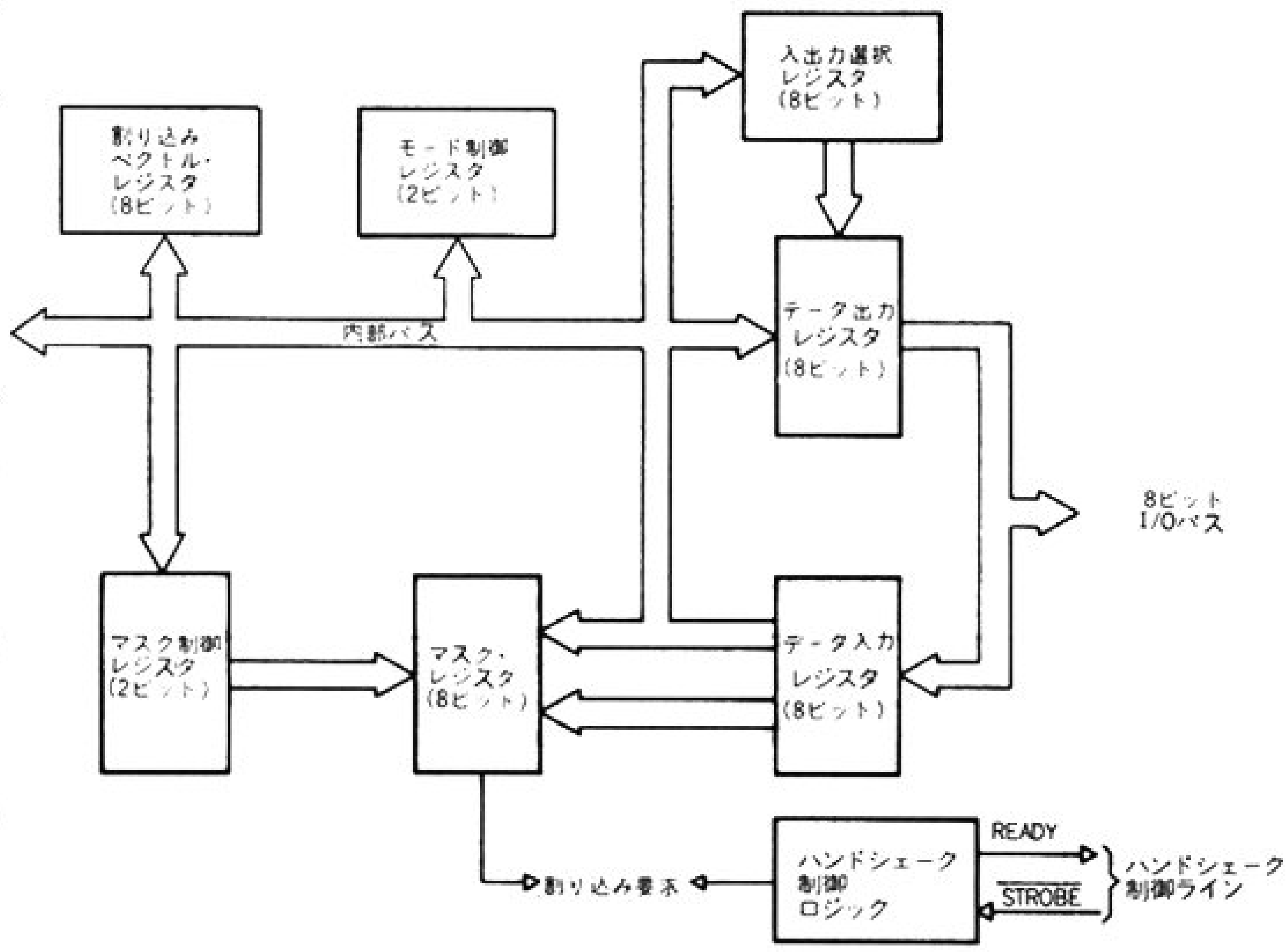
マスク制御レジスタは2ビットで，各ビットにより割り込みを起こすべき状態（“H” または “L”）の指定と，割り込み条件（ANDまたはOR）の指定を行います．この条件がANDであると，マスクのかかっていないすべてのピンが割り込みを発生する状態になったときに，CPUに対して割り込み要求が出され，条件がORの場合にはマスクのかかっていないピンのどれか一つでも割り込みを発生すべき状態になると，CPUに対して割り込み要求が出されます．

PIOはポートA，ポートBそれぞれが割り込みベクトル・レジスタを持っています．PIO内部には前項で簡単に述べたデイジー・チェインがしかれており，ポートA，ポートB間の割り込み優先順位制御が行われています．なお順位はポートA，ポートBの順です．

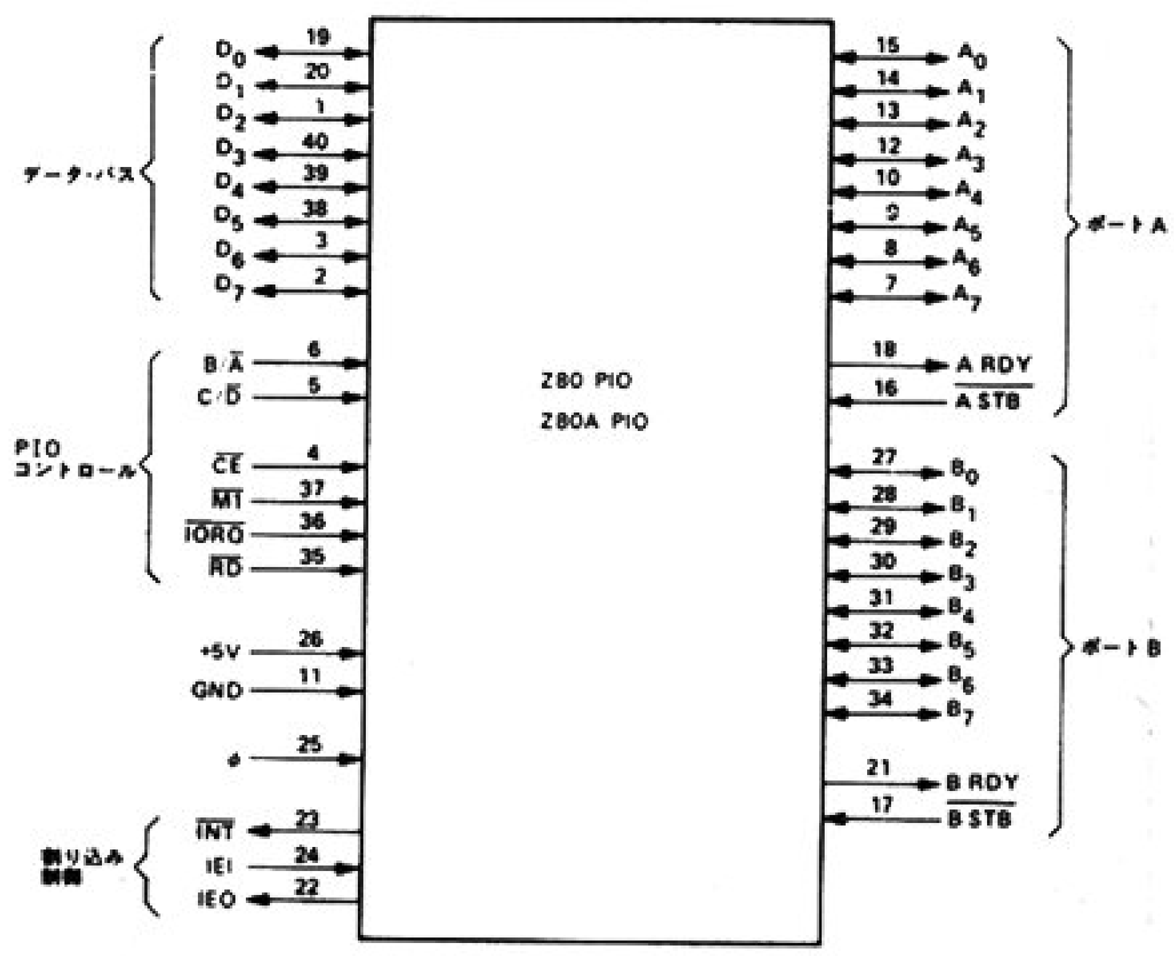
図3にPIOのピン接続図を，**図4**に各ピンの機能を示します．また**図5**にPIOのタイミングを，**図6**に電気的特性を示します．

図7にZ80　PIOのプログラミング・モデルを示します．Z80　PIOは2ポート(ポートA，ポートB)構成であり，各ポートには2本（レディおよびストローブ)のハンド・シェーク・ラインが付属しています．図

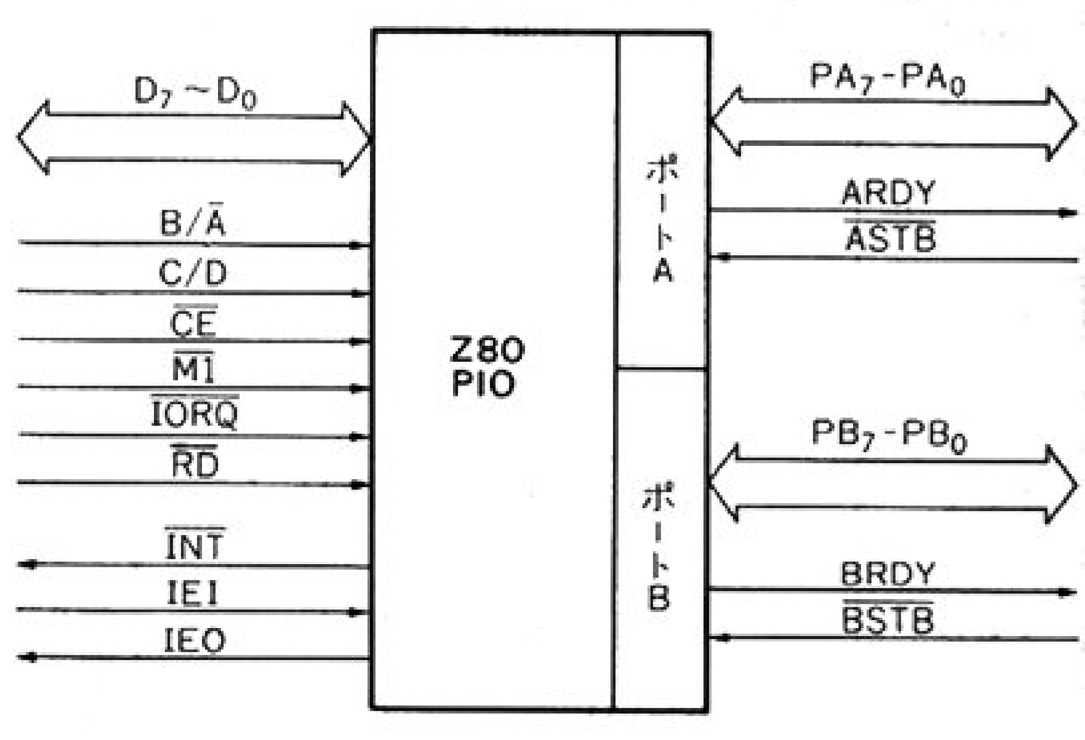
〈図2〉　ポートA，Bのブロック・ダイヤグラム



〈図3〉　Z80 PIOのピン接続図



〈図7〉　Z80　PIOのプログラミング・モデル



B/$\overline{A}$	C/$\overline{D}$	セレクトされるもの
0	0	ポートAデータ
0	1	ポートAコントロール
1	0	ポートBデータ
1	1	ポートBコントロール

〈図8〉
Z80 PIO のアドレッシング

($\overline{CE}$="L"のとき)

8にZ80 PIOの内部レジスタのアドレッシングを示します．ポートA／ポートBの選択はB／$\overline{A}$により行い，データ・レジスタ／コントロール・レジスタの選択はC／$\overline{D}$により行います．これら二つの信号としてはCPUのアドレス・ライン A_1，A_0 がよく使われます．

Z80 PIOの動作モード

Z80 PIOには四つの動作モードがあります．

❶モード0（出力モード）

〈図4〉

ピン名	ピン番号	I/O	信号名	説明
D_0～D_7	19, 20, 1 40, 39, 38 3, 2	I/O	Z80 CPU Data Bus	双方向性，3ステートのZ80 CPUバス． Z80 CPUとPIO間のデータおよびコマンドの授受はこのデータ・バスを介して行われる．D_0がLSB．
B/$\overline{A}$	6	I	Port B or A Select	ポート・セレクト信号． このピンの状態により，Z80 CPUとPIO間で授受されるデータまたはコマンドが対象とするポートを規定する． H：ポートB選択 L：ポートA選択
C/$\overline{D}$	5	I	Control or Data Select	コントロール/データ選択信号． このピンの状態により，B/$\overline{A}$で選択されたポートのコントロール・ポート，またはデータ・ポートを選択する． B/$\overline{A}$ ｜ C/$\overline{D}$ ｜ 選択されるポート L ｜ L ｜ ポートA・データ L ｜ H ｜ ポートA・コントロール H ｜ L ｜ ポートB・データ H ｜ H ｜ ポートB・コントロール
$\overline{CE}$	4	I	Chip Enable	チップ・イネーブル信号． このピンが"L"であるとPIOはイネーブルされる．通常はI/Oアドレス・デコーダの出力を接続する．
ϕ	25	I	System Clock	システム・クロック． 通常はCPUクロックϕを用いる． Z80 PIO　2.5MHz (max)　Z80B PIO　6 MHz (max) Z80A PIO　4 MHz (max)
$\overline{M1}$	37	I	Machine Cycle One	CPUの$\overline{M1}$信号を接続する（"L"アクティブ）． PIOは$\overline{M1}$により割り込み制御ロジックの同期化（CPUとの）をはかる． $\overline{IORQ}$または$\overline{RD}$を"H"にしてこの$\overline{M1}$を"L"（最低2クロック期間）にすると，PIOはリセットされる．
$\overline{IORQ}$	36	I	Input Output Request	CPUの$\overline{IORQ}$信号を接続する（"L"アクティブ）． この信号はB/$\overline{A}$，C/$\overline{D}$，$\overline{CE}$ならびに$\overline{RD}$信号と関連して，CPUとPIO間のデータの授受を制御する．$\overline{CE}$，$\overline{RD}$，$\overline{IORQ}$が"L"であると，B/$\overline{A}$により選択されたポートのデータがCPUに転送される．また$\overline{CE}$，$\overline{IORQ}$が"L"であると，B/$\overline{A}$により選択されたポートにデータまたはコマンドが書き込まれる．
$\overline{RD}$	35	I	Read	CPUの$\overline{RD}$信号を接続する（"L"アクティブ）． この信号はB/$\overline{A}$，C/$\overline{D}$，$\overline{CE}$，$\overline{IORQ}$と関連して，CPUとPIO間で転送されるデータの方向を制御する．
IEI	24	I	Interrupt Enable In	割り込みデイジー・チェイン用の信号で，この信号が"H"であるときに限りPIOはCPUのINTAサイクルに応じることができる．
IEO	22	O	Interrupt Enable Out	割り込みデイジー・チェイン用の信号で，IEIが"H"でかつPIOが割り込み要求を持っていないときのみ"H"となる．IEIが"L"であるとき，またはPIOが割り込み要求を持っているときには"L"となる．

❷モード1(入力モード)
❸モード2(双方向モード)
❹モード3(ビット・モード)

これら四つのうち,モード3のビット・モードが一番わかりやすいのでこれから説明します.

・モード3(ビット・モード)

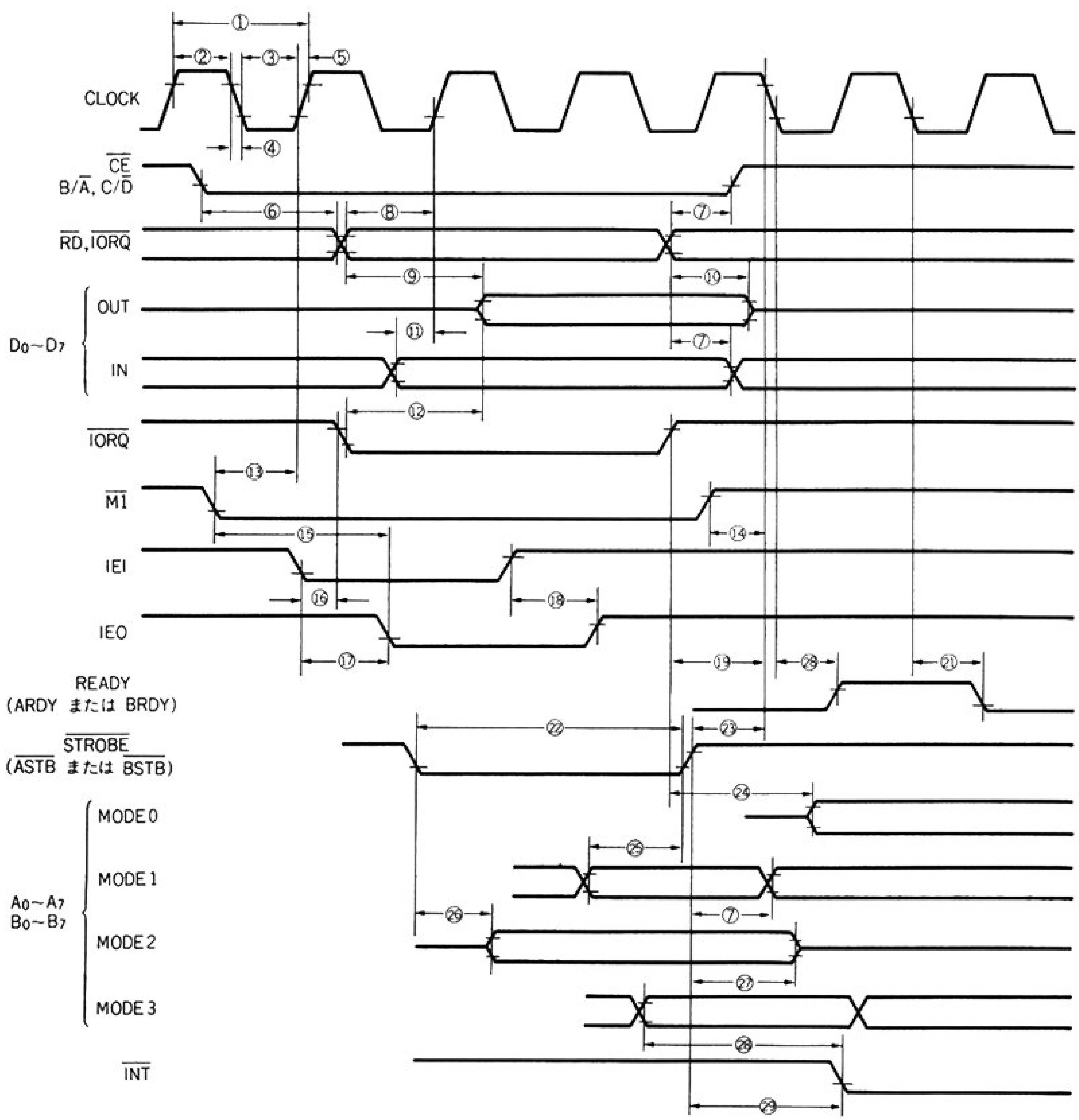
このモードではポートの各ビットに対して"入力"または"出力"の指定を行うことができます.CPUからモード3のポートに対してデータを出力した場合,"出力"と指定されたビットに対してのみデータが書き込まれます.

また,CPUからモード3のポートに対してリード命令を出した場合には,"入力"と指定されたビットに対応するPIOのピンの状態と,"出力"と指定されたビットに対応する内部データ・ラッチの状態がCPUに対して送られます.したがって,ポートの全ビットを"出力"と指定して,このポートに対してデータをライトし,次にそのポートからデータをリードすると,ライトしたデータと同じものがCPUに対して送られます.

PIOの各ピンの機能

ピン名	ピン番号	I/O	信号名	説明
$\overline{INT}$	23	O	Interrupt Request	CPUの$\overline{INT}$信号と接続する.この信号を"L"にすることにより,PIOはCPUに対して割り込み要求を行う.オープン・ドレイン型であるのでプル・アップ抵抗により,いくつかの周辺LSIの$\overline{INT}$信号をワイヤードORすることができる.
A_0~A_7	15−12 10−7	I/O	Port A Bus	ポートAデータ・バス.このバスを介してPIOと周辺装置はデータの授受を行う.A_0がLSBである.
$\overline{ASTB}$	16	I	Port A Strobe	ポートAストローブ. この信号の意味はポートAの動作モードにより異なる. **1)バイト出力モード**:このストローブの立ち上がりにより,周辺装置はPIOからデータを受け取ったことを示す. **2)バイト入力モード**:このストローブの立ち上がりにより,周辺装置はPIOのポートA入力データ・レジスタにデータをロードする. **3)双方向性モード**:このストローブが"L"のときにポートA出力データ・レジスタの内容がA_0~A_7に出力される. **4)ビット・モード**:この信号は使用されない.
ARDY	18	O	Register A Ready	レジスタAレディ. この信号の意味もポートAの動作モードにより異なる. **1)バイト出力モード**:この信号が"H"になると,ポートAのデータ出力レジスタにデータがロードされ,A_0~A_7の状態が安定となり,周辺装置に対してデータを転送できることを示す. **2)バイト入力モード**:この信号が"H"であると,ポートAのデータ入力レジスタが空であり,次のデータを受け取る用意のあることを示す. **3)双方向性モード**:この信号は,ポートA出力データ・レジスタにデータが準備されていることを示す.このモードでは$\overline{ASTB}$が"L"になるまで,データはA_0~A_7上に乗らない. **4)ビット・モード**:この信号は使用されない.
B_0~B_7	27−34	I/O	Port B Bus	ポートBデータ・バス. このバスの機能はA_0~A_7と同じであるが,このバスは1.5Vで1.5mAを供給できるので,ダーリントン・トランジスタをドライブできる. B_0がLSBである.
$\overline{BSTB}$	17	I	Port B Strobe	ポートBストローブ. この信号の機能は$\overline{ASTB}$のそれと同じであるが,次の違いがある. ポートAが双方向性モードのときは,この信号は周辺装置のデータをポートA入力データ・レジスタにロードするのに用いられる.
BRDY	21	O	Register B Ready	レジスタBレディ. この信号の機能はARDYのそれと同じであるが,次の違いがある. ポートAが双方向性モードのときは,この信号はポートA入力データ・レジスタが空で,次のデータを受け取る用意のあることを示す.

〈図 5〉 Z80



記 号	パラメータ	最小値	最大値	単位	テスト条件
V_{ILC}	クロック入力の"L"電圧	−0.3	+0.45	V	
V_{IHC}	クロック入力の"H"電圧	$V_{CC}-0.6$	$V_{CC}+0.3$	V	
V_{IL}	入力"L"電圧	−0.3	+0.8	V	
V_{IH}	入力"H"電圧	+2.0	V_{CC}	V	
V_{OL}	出力"L"電圧		+0.4	V	$I_{OL}=2mA$
V_{OH}	出力"H"電圧	+2.4		V	$I_{OH}=-250\mu A$
I_{LI}	入力漏れ電流		±10	μA	$V_{IN}=0\sim V_{CC}$
I_{LO}	フロート時の出力漏れ電流		±10	μA	$V_{OUT}=0.4\sim V_{CC}$
I_{CC}	電源電流		100.0	mA	$V_{OH}=1.5V$
I_{OHD}	ダーリントン駆動電流	−1.5		mA	$R_{EXT}=390\Omega$

〈図 6〉

Z80 PIO の電気的特性

記 号	パラメータ	最大値	単位
C_{CLK}	クロック入力容量	10	pF
C_{IN}	入力容量	5	pF
C_{OUT}	出力容量	10	pF

PIOのタイミング

番号	記　号	パラメータ	Z80-PIO		Z80A-PIO		Z80B-PIO		注
			最小値	最大値	最小値	最大値	最小値	最大値	
1	T_{cC}	クロックのサイクル・タイム	400	(1)	250	(2)	165	(1)	
2	T_{wCh}	クロックの幅("H")	170	2000	105	2000	65	2000	
3	T_{wCl}	クロックの幅("L")	170	2000	105	2000	65	2000	
4	T_{fC}	クロックの立ち下がり時間		30		30		20	
5	T_{rC}	クロックの立ち上がり時間		30		30		20	
6	$T_{sCS(RI)}$	$\overline{\text{RD}}$, $\overline{\text{IORQ}}$に対する$\overline{\text{CE}}$, $\overline{\text{BA}}$のセットアップ時間	50		50			50	(6)
7	T_h	ホールド・タイム	0		0		0		
8	$T_{sRI(C)}$	クロックの立ち上がりに対する$\overline{\text{RD}}$, $\overline{\text{IORQ}}$のセットアップ時間	115		115		70		
9	$T_{dRI(DO)}$	$\overline{\text{RD}}$, $\overline{\text{IORQ}}$の立ち下がりに対するデータ出力遅延時間		430		380		300	(2)
10	$T_{dRI(DOs)}$	$\overline{\text{RD}}$, $\overline{\text{IORQ}}$の立ち上がりに対するデータ出力フロート遅延時間		160		110		70	
11	$T_{sDI(C)}$	クロックの立ち上がりに対するデータ入力セットアップ時間	50		50		50		$C_L=50$pF
12	$T_{dIO(DOI)}$	$\overline{\text{IORQ}}$の立ち下がりに対するデータ出力遅延時間(INTAサイクル)		340		160		120	(3)
13	$T_{sM1(Cr)}$	クロックの立ち上がりに対するM1のセットアップ時間	210		90		70		
14	$T_{sM1(Cf)}$	クロックの立ち下がりに対するM1のセットアップ時間(M1サイクル)	0		0		0		(8)
15	$T_{dM1(IEO)}$	M1に対するIEOの遅延時間		300		190		100	(5, 7)
16	$T_{sIEI(IO)}$	$\overline{\text{IORQ}}$の立ち下がりに対するIEIのセットアップ時間(INTAサイクル)	140		140		100		(7)
17	$T_{dIEI(IEOf)}$	IEIの立ち下がりに対するIEOの遅延時間		190		130		120	(5)
18	$T_{dIEI(IEOr)}$	IEIの立ち上がりに対するIEOの遅延時間		210		160		160	(5)
19	$T_{cIO(C)}$	クロックの立ち下がりに対する$\overline{\text{IORQ}}$のセットアップ時間	220		200		170		
20	$T_{dC(RDYr)}$	クロックの立ち下がりに対するREADYの遅延時間		200		190		170	(5)
21	$T_{dc(RDYf)}$	クロックの立ち下がりに対するREADYの遅延時間		150		140		120	(5)
22	T_{wSTB}	$\overline{\text{STROBE}}$のパルス幅	150		150		120		(4)
23	$T_{sSTB(C)}$	クロックの立ち下がりに対する$\overline{\text{STROBE}}$のセットアップ時間	220		220		150		(5)
24	$T_{dIO(PD)}$	$\overline{\text{IORQ}}$の立ち上がりに対するポート・データの遅延時間(モード0)		200		180		160	(5)
25	$T_{sPD(STB)}$	$\overline{\text{STROBE}}$の立ち上がりに対するポート・データのセットアップ時間(モード1)	260		230		190		
26	$T_{dSTB(PD)}$	$\overline{\text{STROBE}}$の立ち下がりに対するポート・データの安定時間(モード2)		230		210		180	(5)
27	$T_{dSTB(PDr)}$	$\overline{\text{STROBE}}$の立ち上がりに対するポート・データのフロート遅延(モード2)		200		180		160	$C_L=50$pF
28	$T_{dPD(INT)}$	ポート・データの一致に対する$\overline{\text{INT}}$の遅延時間(モード3)		540		490		430	
29	$T_{dSTB(INT)}$	$\overline{\text{STROBE}}$の立ち上がりに対する$\overline{\text{INT}}$の遅延時間		490		440		350	

(単位：ns)

【注】

(1) $T_cC = T_{wCh} + T_{wCl} + T_{rC} + T_{fC}$

(2) 負荷を50pF増すごとに(最高200pF) $T_{dRI(DO)}$は10ns増える.

(3) 負荷を50pF増すごとに(最高200pF) $T_{dIO(DOI)}$は10ns増える.

(4) モード2では$T_{wSTB} > T_{sPD(STB)}$.

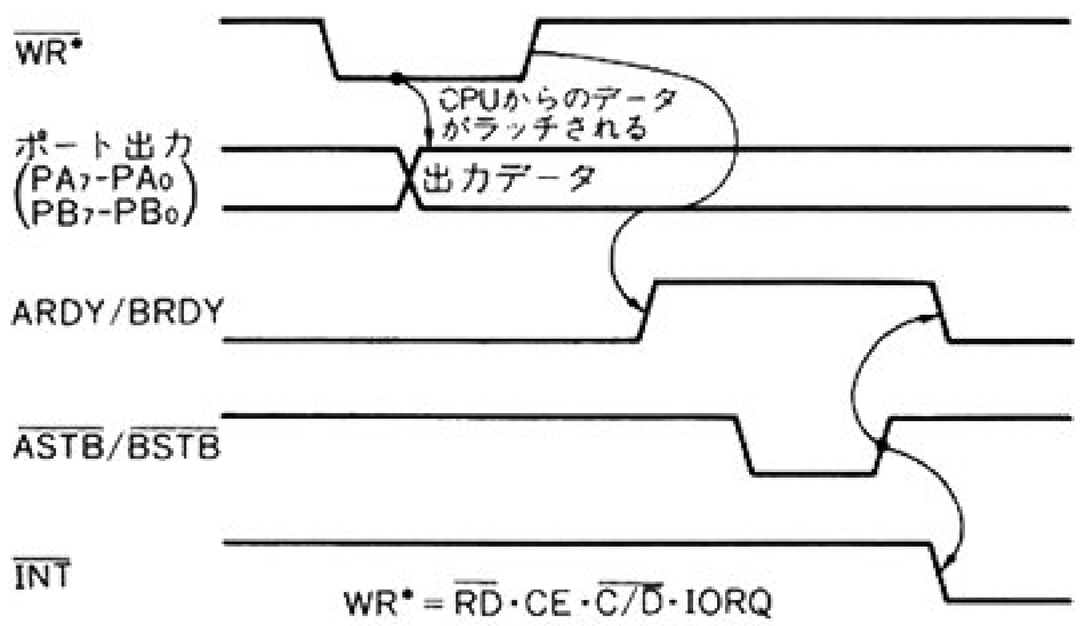
(5) 負荷を10pF増すごとに(最高100pF)2ns増える.

(6) $T_{sCS(RI)}$の値は減らしてもよいが, 減らした分だけ$T_{dRI(DO)}$に加える必要がある.

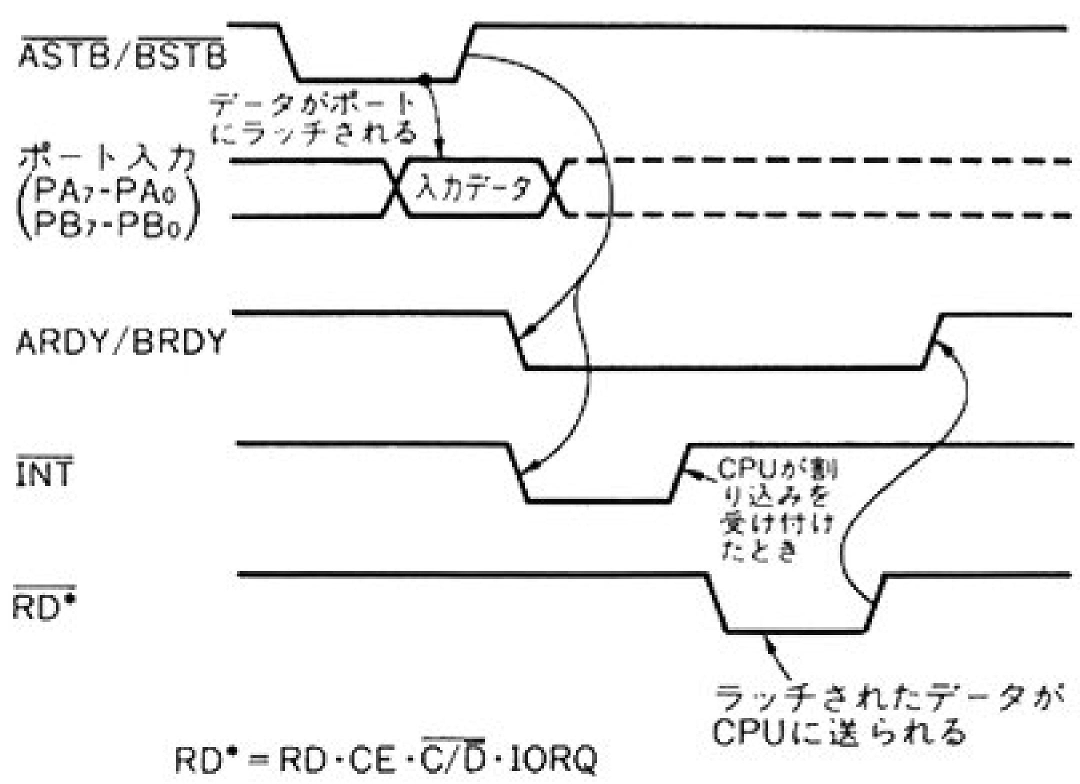
(7) $2.5T_cC > (N-2)T_{dIEI(IEOf)} + T_{dM1(IEO)} + T_{sIEI(IO)}$ + TTLの遅延時間.

(8) PIOをリセットするには, $\overline{\text{M1}}$を最低2クロック期間"L"にする必要がある.

〈図9〉 Z80 PIO モード0タイミング．$\overline{INT}$が“L”になるのは PIO のの割り込みがイネーブルされているときだけ



〈図10〉 Z80 PIO モード1タイミング．$\overline{INT}$が“L”になるのは PIO の割り込みがイネーブルされているときだけ



Z80 PIO のモード3ポートに対するデータ・リードに関する良い点は，$\overline{RD}$ 信号の立ち下がりでポートの入力状態が内部にラッチされることです．このおかげでいつ変化するかわからないような入力状態を読み込む場合でも，CPU の $\overline{RD}$ 中にデータが変化するようなことはありません．このことはデータ・バス・パリティを付加するような場合には重要です．

モード3では“入力”と指定されたビットのピンの状態が“H”または“L”になるのを PIO が監視し，割り込みを発生する機能があります．この場合，いくつかのビットの中の一つでも割り込み発生の状態になれば，割り込みを発生したり（OR 機能），いくつかのビットがすべて割り込み発生の状態になったときにのみ割り込みを発生させたりする（AND 機能）ことができます．

なおモード3ではハンドシェーク・ラインは使用されず，レディ(ARDY, BRDY)・ラインは常に“L”になります．

・モード0（出力モード）

モード0ではポートにデータをライトすると，そのデータが PIO 内部にラッチされ，ピン上に出力されます．その後レディ(ARDY または BRDY)・ラインが“H”になり，外部に対してデータの用意ができたことを知らせます．

これに対して外部からストローブ（$\overline{ASTB}$ または $\overline{BSTB}$）を“L”にすると，その立ち上がりでレディ・ラインは再度“L”になります．このときポートの割り込みがイネーブルされていれば，割り込み要求線，$\overline{INT}$ が“L”になります．図9にモード0のタイミングを示します．

・モード1（入力モード）

図10にモード1のタイミングを示します．モード1の動作は外部からストローブ($\overline{ASTB}$／$\overline{BSTB}$)が入力されることにより始まります．ストローブが“L”になるとポートのピンの状態が PIO 内部にラッチされ，次にレディ(ARDY／BRDY)が“L”になり，インプット・バッファがビジーであることを外部に示します．

また，このときポートの割り込みがイネーブル状態であれば，$\overline{INT}$ が“L”になり外部からデータの入力があったことを CPU に対して示します．ここで CPU がポート・データをリードすると，再度レディ・ラインは“H”になり，インプット・バッファが空であることを外部に知らせます．

・モード2（双方向モード）

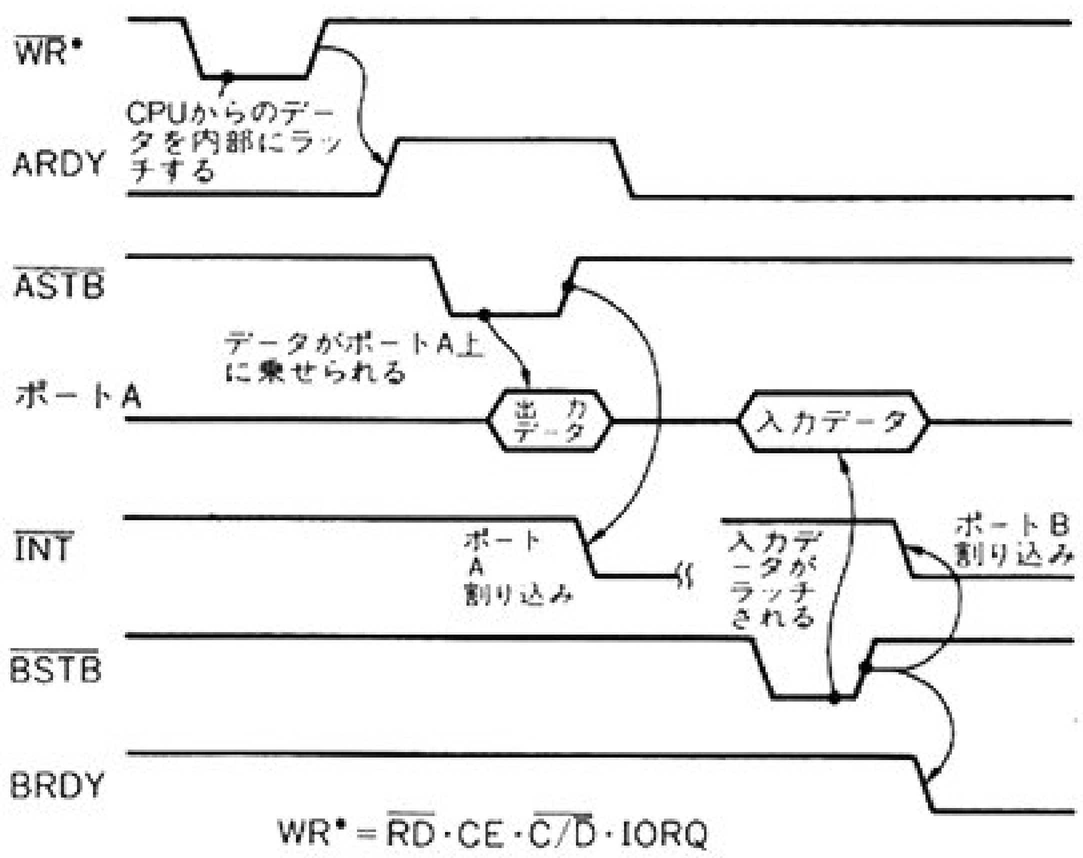
このモードはポートAに対してのみ有効であり，ポートAの制御にポートBの BRDY および $\overline{BSTB}$ も使用しますので，ポートAをモード2に設定したときには，ポートBはモード3でしか動くことができません．とはいえポートAをモード2で動かすにはレディとストローブ以外に制御線を何本か使用するのがふつうであり，この制御線としてポートBを用いればよいので何の不自由も感じないのが実情です．

図11にモード2のタイミングを示します．この図を見ればわかるように，モード2は図9のモード0と図10のモード1のタイミングを重ね合わせたようなものであり，異なるのはポートAの状態が双方向性であり，ストローブが入力されたときにのみ，ポートAにデータが乗る点です．

では図11を見ながらモード2の動作を説明しましょう．まず CPU からデータがポートAに対してライトされると，データが PIO 内部のアウトプット・バッファにラッチされ，ARDY が“H”になり出力データが用意できていることを外部に知らせます．

これに対して外部から $\overline{ASTB}$ が入力されると，ラッチされていたデータがポートA上に乗せられます．

〈図11〉 Z80 PIOモード2タイミング．$\overline{INT}$が“L”になるのは，PIOの割り込みがイネーブルされているときだけ



ARDYは$\overline{ASTB}$の立ち上がりにて再度“L”になり，このときポートAの割り込みがイネーブルされていれば$\overline{INT}$が“L”になり（ポートA割り込み），アウトプット・バッファが空になったことをCPUに知らせます．

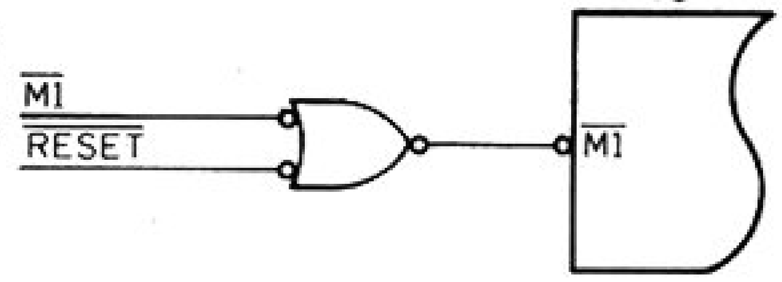
外部からの入力データ・シーケンスはデータをポートA上に乗せ，$\overline{BSTB}$を“L”にすることから始まります．$\overline{BSTB}$が“L”になるとPIOはポートA上のデータをインプット・バッファにラッチし，BRDYを“L”にしてインプット・バッファがビジーであることを外部に示します．

また，このときポートBの割り込みがイネーブルされていれば，$\overline{INT}$が“L”になり（ポートB割り込み），入力データが用意できていることをCPUに知らせます．図11には示されていませんが，BRDYはCPUがポートAのデータをリードすることにより“H”に戻ります．

このようにモード2のARDYは出力データの用意があることを外部に示し，BRDYはインプット・バッファが空であることを外部に示します．またポートAの割り込みはアウトプット・バッファが空であることをCPUに知らせ，ポートB割り込みはCPUに対して入力データの用意があることを知らせます．

PIOをモード2でかつ割り込み駆動型（Interrupt Driven）で使用する場合にはA，B両ポートの割り込みを使用するので，ビット・モードのポートBに割り込みを発生させると，ポートB割り込みの要因を識別することが困難な場合もあります．このような場合にはポートBに対して割り込みマスクをセットして，ビット・モードの割り込みを起こさせないようにする必要があります．

〈図12〉
外部信号（$\overline{RESET}$）によるPIOのリセット



・PIOのリセット

ＰＩＯにはリセット端子はありませんが，パワーオン・クリヤ回路が内蔵されており，電源を投入するとＰＩＯはリセットされます．ＰＩＯがリセットされると次のような状態になります．すなわち，

(1) A，B両ポートのマスク・レジスタはすべてセットされ，割り込み禁止状態になります．
(2) A，B両ポートのデータ・バスはハイ・インピーダンスにされ，ハンドシェーク信号，レディは“L”になります．そしてモード1がプリセットされます．
(3) 割り込みベクトル・レジスタの状態は変化しません．
(4) 各ポートの割り込みイネーブルFFはクリヤされます．
(5) 各ポートのデータ出力レジスタはゼロにリセットされます．

パワーオン・クリヤのほかにＰＩＯは$\overline{RD}$および$\overline{IORQ}$を“H”にして，$\overline{M1}$だけを単独に“L”にすることにより，クリヤすることができます．したがって，図12のようにゲートを一つ設けるだけで，任意の時点にPIOをクリヤすることができます．ただし，PIOをリセットするには最低2クロック期間$\overline{M1}$を“L”にする必要があります．

なおPIOにリセット入力がないのはピンの数に制限があるためです．

Z80 PIOのプログラミング

Z80 PIOに対して所望の動作を行わせるためには，以下に説明するコントロール・ワードをポートのコントロール・レジスタにロードする必要があります．コントロール・レジスタは$C/\overline{D}=1$によりアクセスされます．

・動作モードの設定

Z80 PIOのモード・ワードを図13に示します．このモード・ワードの下位4ビットは常に“1”であり，上位2ビットにより四つのモードのうちの一つを選択します．ビット5とビット4は“Don't Care”であり，値は何であってもかまいません．

モード3のビット・モードを選択した場合には，次のコントロール・ワードは図14に示すデータ・ディレクション・ワードでなければなりません．このワード

〈図13〉 Z80 PIOのモード・ワード．モード・ワードのビット3～0は常に"1"であり，ビット5, 4はDon't Careである

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
M_1	M_0	×	×	1	1	1	1

M_1	M_0	
0	0	モードO(出力モード)
0	1	モード1(入力モード)
1	0	モード2(双方向モード)
1	1	モード3(ビット・モード)

モード・ワードの値	セレクトされるモード
$0F_{16}$	モード0
$4F_{16}$	モード1
$8F_{16}$	モード2
CF_{16}	モード3

〈図14〉 Z80 PIOのデータ・ディレクション・ワード

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
I/O_7	I/O_6	I/O_5	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

$I/O_n=0$　ビットn＝出力
$I/O_n=1$　ビットn＝入力

PIOのポートをモード3（ビット・モード）に設定するモード・ワード（CF_{16}）を送った直後のコントロール・ワードは，このデータ・ディレクション・ワードでなければならない．

はモード3に指定されたポートの各ビットの入／出力を規定します．このワードのビットnが"0"であると，ポートのビットnは出力となり，ビットnが"1"であると，ポートのビットnは入力となります．

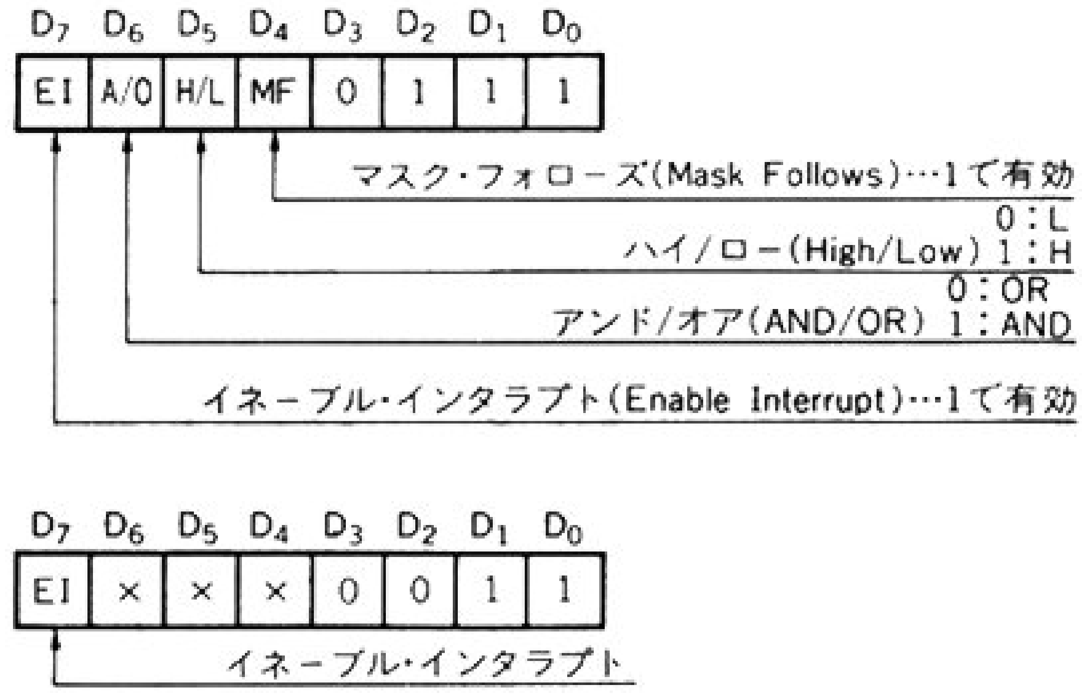
・インタラプト・コントロール

図15にZ80 PIOのインタラプト・コントロール・ワードを示します．インタラプト・コントロール・ワードは2種類あります．まず下位4ビットが"0111"のコントロール・ワード(図15の上)について説明します．このワードのビット7はイネーブル・インタラプト・フラグであり，このビットが"1"であると当該ポートの割り込みは"可"となり，"0"であると"否"となります．

このワードのビット6～4はモード3においてのみ有効です．ビット4が"1"であると「このコントロール・ワードの後に図16のインタラプト・マスク・ワードがロードされるぞ」という宣言になります．

インタラプト・マスク・ワードは"入力"と指定されたビットのうち，割り込みの対象とならないビットを指定するワードです．$MB_n=1$であるとビットnに対してマスクがかかり，そのビットは割り込みの対象から除外されます．なおPIOはリセットされると全ビットに対してマスクを"1"にした状態になります．

〈図15〉 Z80 PIOのインタラプト・コントロール・ワード．下のコントロール・ワードは，PIO中のインタラプト・イネーブルFFだけをセット／リセットするのに用いられる



〈図16〉 Z80 PIOのインタラプト・マスク・ワード

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

$MB_n=0$の場合，ビットnがインタラプトの対象となる

〈図17〉 Z80 PIOのインタラプト・ベクトル

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0

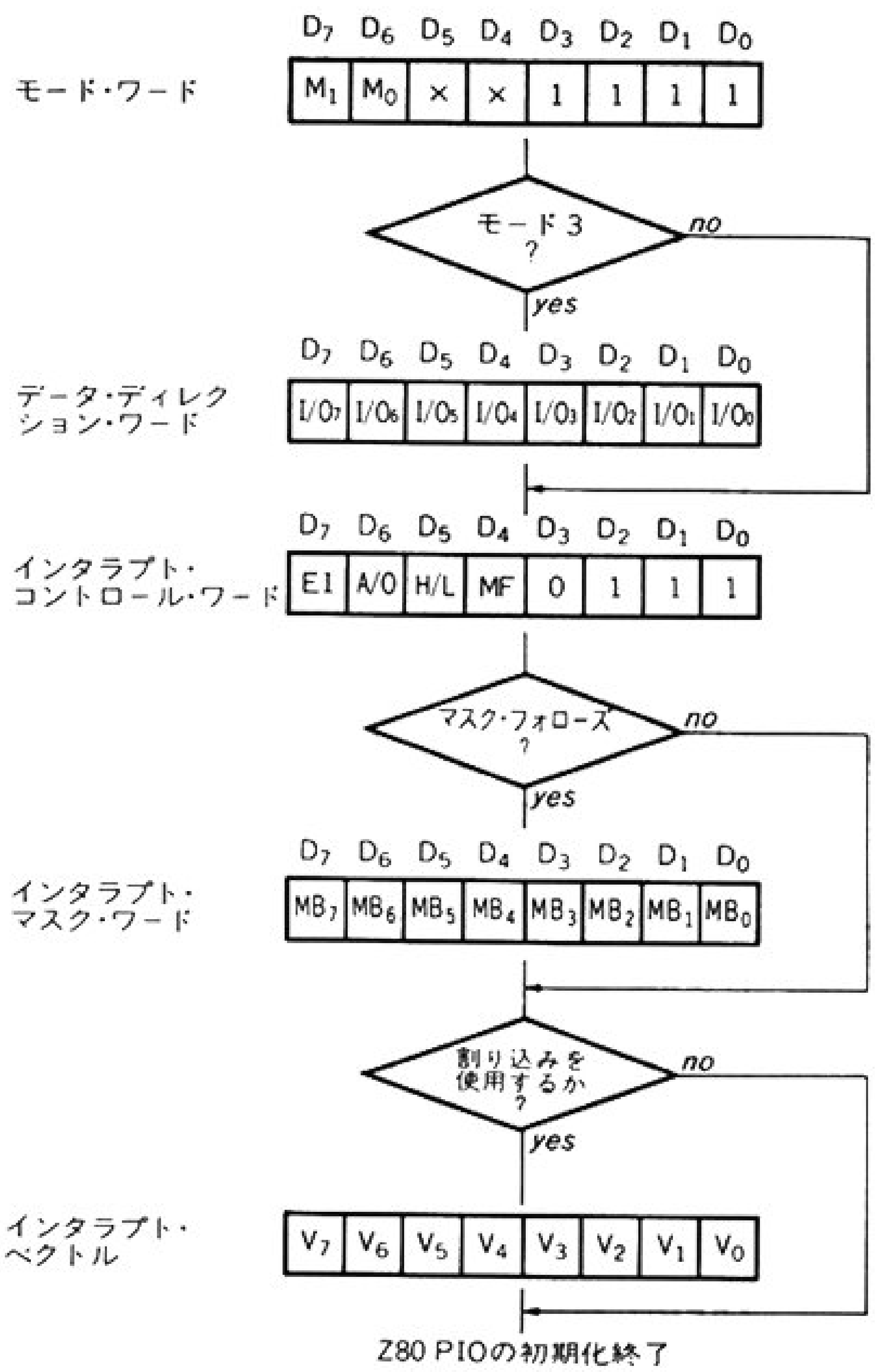
ポートA，ポートBに対して独立のベクトルを設定できる

インタラプト・コントロール・ワードのビット5は，マスクのかかっていないビットの割り込みを"H"で有効にするのか，"L"で有効にするのかの指定を行います．このビットが"1"であると"H"が有効であり，"0"であると"L"が有効となります．

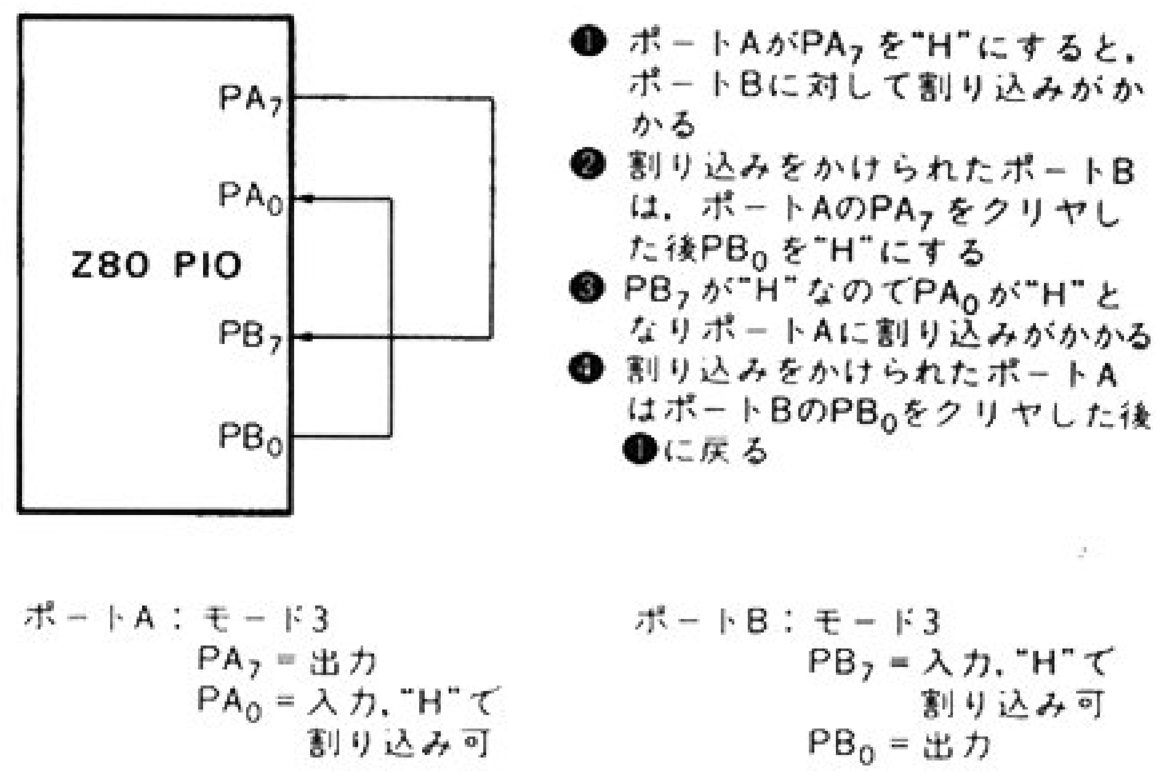
このワードのビット6はマスクのかかっていないすべてのビットが有効極性となった場合に割り込みを発生するのか（AND機能），あるいは1ビットでも有効極性のビットがあれば割り込みを発生させるのか(OR機能）の指定を行います．ビット6が"1"であるとAND機能が働き，"0"であるとOR機能が働きます．

次に下位4ビットが"0011"であるインタラプト・コントロール・ワード（図15の下）について説明します．このワードは割り込みに関する他の条件（マスク，AND／OR，ハイ／ロー）に影響を与えることなく，PIO内部のインタラプト・イネーブル・フリップフロップだけをセット／リセットするのに用いられます．

〈図18〉 Z80 PIOのプログラミング・フロー



〈図19〉 PIOをモード3で動かす例



・インタラプト・ベクトルのロード

図17にZ80 PIOのインタラプト・ベクトルを示します．PIOはポートA，ポートBそれぞれがインタラプト・ベクトル・レジスタを持っていますので，各ポートに対して独立のベクトルを設定できます．

以上のZ80 PIOのプログラミングのまとめとして，図18にプログラミング・フローを示します．

Z80 PIOのプログラム例

・モード3（ビット・モード）

モード3のプログラム例として図19の構成を考えます．ポートAおよびポートBは両方ともモード3とし，ポートAのビット7（PA_7）は出力，ビット0（PA_0）は入力とし，PA_0が"H"のときポートAは割り込み可とします．また，ポートBのビット7（PB_7）は入力，ビット0（PB_0）は出力とし，PB_7が"H"のときポートBは割り込み可とします．

プログラムはまずPA_7を"H"とします．これによりPB_7が"H"となるので，ポートBに割り込みがかかります．ポートBの割り込みルーチンはPA_7をクリヤした後にPB_0を"H"にします．PB_0が"H"になると今度はポートAの割り込みが発生し，ポートAの割り込みルーチンはPB_0をクリヤした後にPA_7を"H"にします．以上の過程を永遠に続けることにします．

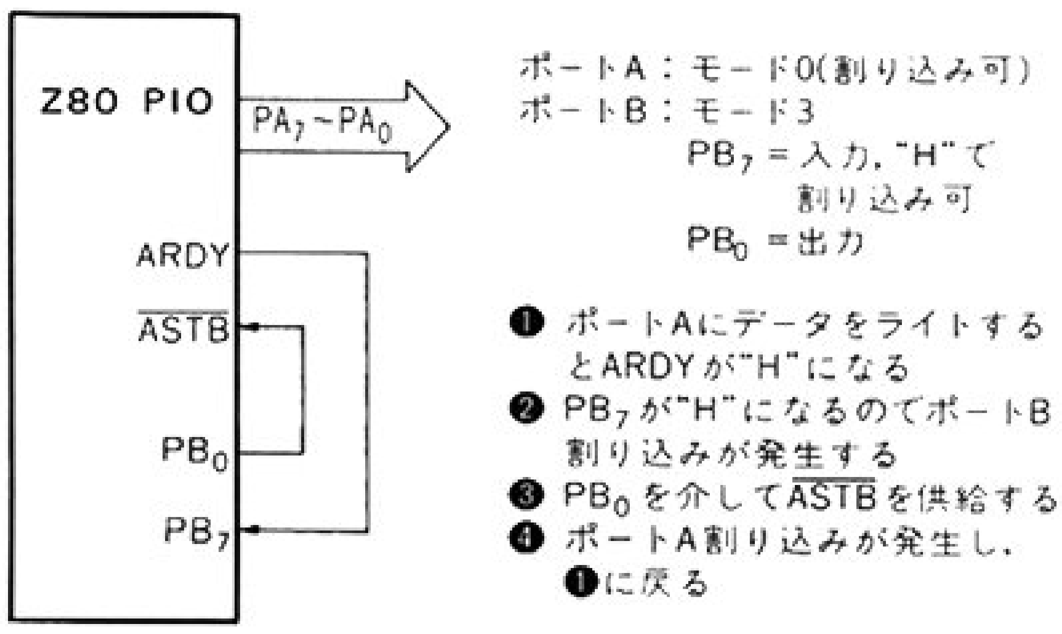
Prog.❶に以上の動作を行うプログラムを示します．

〔Prog.❶〕 Z80 PIOモード3 （図19の構成で動作する）

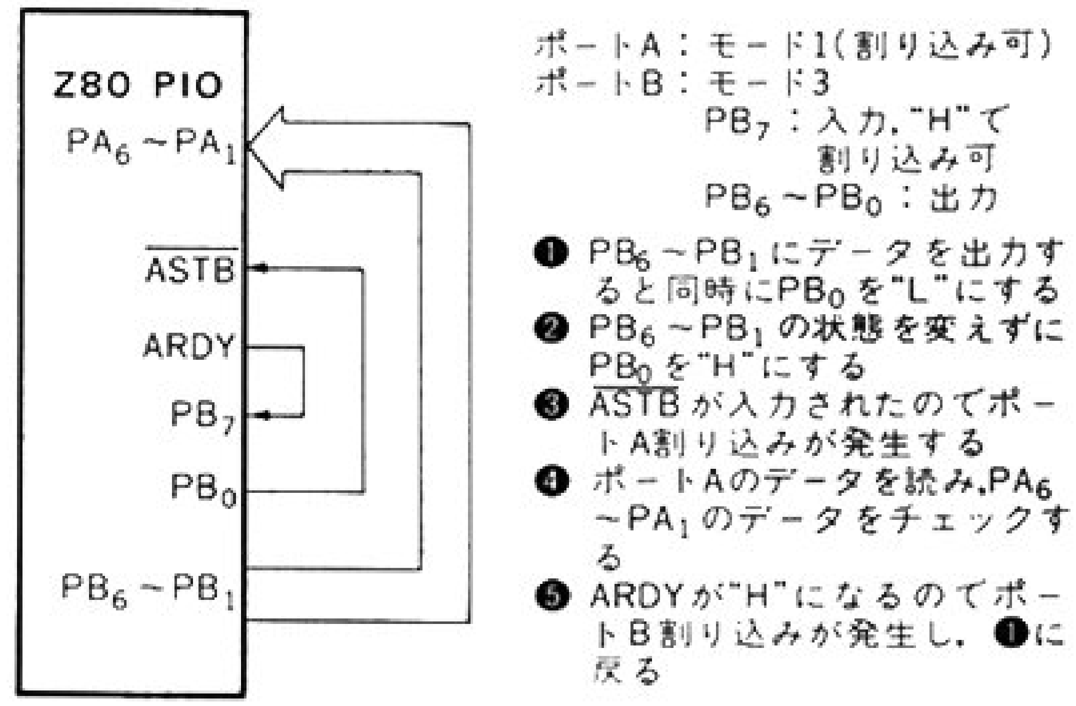
```
                ;
                ;
                ;       Z80 PIO MODE 3 PROGRAMMING
                ;
0000            PIOAD:  EQU   0                    ;PIO PORT A DATA
0001            PIOAC:  EQU   1                    ;PIO PORT A CONTROL
0002            PIOBD:  EQU   2                    ;PIO PORT B DATA
0003            PIOBC:  EQU   3                    ;PIO PORT B CONTROL
                ;
1000            STACK:  EQU   1000H                ;STACK
                ;
                ;
0000 310010     BEGIN:  LD    SP,STACK             ;SET STACK POINTER
0003 3E01               LD    A,01H                ;SET IR
0005 ED47               LD    I,A
0007 ED5E               IM    2                    ;SET MODE 2 INTERRUPT
```

```
                   ;
                   ;         PIO INITIALIZATION
0009 211F00                  LD    HL,PACMD              ;LOAD PIO PORT A COMMAND ADDRESS
000C 0605                    LD    B,PACMDL-PACMD        ;LOAD PIO PORT A COMMAND LENGTH
000E 0E01                    LD    C,PIOAC               ;LOAD PIO PORT A CONTROL ADDRESS
0010 EDB3                    OTIR                        ;LOAD COMMAND TO PIO PORT A
0012 0605                    LD    B,PBCMDL-PBCMD        ;LOAD PIO PORT B COMMAND LENGTH
0014 0E03                    LD    C,PIOBC               ;LOAD PIO PORT B CONTROL ADDRESS
0016 EDB3                    OTIR                        ;LOAD COMMAND TO PIO PORT B
                   ;
                   ;         PIO HAS BEEN INITIALIZED
0018 3E80                    LD    A,80H                 ;SET PA7
001A D300                    OUT   (PIOAD),A
001C FB                      EI                          ;ENABLE INTERRUPT
001D 18FE                    JR    $                     ;LOOP ETERNALLY
                   ;
                   ;         PIO PORT A COMMAND CHAIN
001F CF            PACMD:    DB    0CFH                  ;MODE 3
0020 7F                      DB    07FH                  ;PA7=OUTPUT
0021 B7                      DB    0B7H                  ;INTERRUPT CONTROL WORD
0022 FE                      DB    0FEH                  ;INTERRUPT MASK
0023 00                      DB    AINTV-100H            ;INTERRUPT VECTOR
0024               PACMDL:EQU  $
                   ;
                   ;         PIO PORT B COMMAND CHAIN
0024 CF            PBCMD:    DB    0CFH                  ;MODE 3
0025 FE                      DB    0FEH                  ;PB0=OUTPUT
0026 B7                      DB    0B7H                  ;INTERRUPT CONTROL WORD
0027 7F                      DB    07FH                  ;INTERRUPT MASK
0028 02                      DB    BINTV-100H            ;INTERRUPT VECTOR
0029               PBCMDL:EQU  $
                   ;
                   ;         PORT A INTERRUPT
0029 DB00          PAINT:    IN    A,(PIOAD)             ;READ PORT A
002B CB47                    BIT   0,A                   ;PA0="H" ?
002D CA4D00                  JP    Z,ERROR               ;NO, ERROR
0030 3E00                    LD    A,0                   ;CLEAR PORT B
0032 D302                    OUT   (PIOBD),A
0034 3E80                    LD    A,80H                 ;SET PA7="H"
0036 D300                    OUT   (PIOAD),A
0038 FB                      EI                          ;RE-ENABLE INTERRUPT
0039 ED4D                    RETI
                   ;
                   ;         PORT B INTERRUPT
003B DB02          PBINT:    IN    A,(PIOBD)             ;READ PORT B
003D CB7F                    BIT   7,A                   ;PB7="H" ?
003F CA4D00                  JP    Z,ERROR               ;NO, ERROR
0042 3E00                    LD    A,0                   ;CLEAR PORT B
0044 D300                    OUT   (PIOAD),A
0046 3E01                    LD    A,1                   ;SET PB0="H"
0048 D302                    OUT   (PIOBD),A
004A FB                      EI                          ;RE-ENABLE INTERRUPT
004B ED4D                    RETI
                   ;
004D 76            ERROR:    HLT                         ;ERROR STOP
                   ;
                             ORG   0100H
0100 2900          AINTV:    DW    PAINT                 ;PORT A VECTOR
0102 3B00          BINTV:    DW    PBINT                 ;PORT B VECTOR
                   ;
0104                         END
```

〈図20〉 Z80 PIO をモード０で動作させる構成



〈図21〉 Z80 PIO をモード１で動作させる構成



ザイログ社のマニュアルには記述されていませんが，Z80 PIO のビット・モードの割り込みはレベル・センスではなく，エッジ・センスのようです．

つまりいったん割り込みを発生したビット・モードの入力ラインは，一度そのラインの状態を割り込みを発生しないレベルに落とす必要があります．筆者の実験では，そうしないと **Prog.❶** は動きませんでした．

・モード０（出力モード）

図20にモード０のプログラム例を示すための構成を示します．ポートＡの ARDY はポートＢのビット７ (PB_7) に接続され，ポートＢのビット０ (PB_0) はポートＡの $\overline{ASTB}$ に接続されています．ポートＡはモード０に設定し，ポートＢはモード３に設定します．また，PB_0 は出力とし，PB_7 は入力でありかつ "H" で割り込み可とします．

プログラムは，まず PB_0 を"H"にしておいて ($\overline{ASTB}$＝"H") ポートＡにデータを出力します．すると ARDY が "H" になり，これは PB_7 に接続されているのでポートＢに割り込みがかかります．ポートＢの割り込みルーチンは PB_0 をいったん "L" にし，ついで "H" に戻します．これにより，PB_0 は $\overline{ASTB}$ に接続されていますので，$\overline{ASTB}$ が入力されたことになり，その立ち上がり時にポートＡ割り込みが発生します．

ポートＡの割り込みルーチンはポートＡに対してデータを出力します．そして以上の過程は永遠に続きます．

Prog.❷ に以上の動作をするプログラムを示します．このプログラムではポートＡに出力するデータとして 55H と AAH を交互に使用しています．

・モード１（入力モード）

図21に Z80 PIO をモード１で動作させるための構成図を示します．ポートＡはモード１とし，ポートＢはモード３とします．またポートＢの PB_0 は出力，PB_7 は入力としかつ "H" で割り込み可とします．

プログラムは PB_6～PB_1 にデータを出力し，同時に PB_0 を "L" にします．次に PB_6～PB_1 の状態をそのままにして PB_0 を "H" にします．PB_0 はポートＡの $\overline{ASTB}$ に接続されていますので，ポートＡ割り込みが発生します．

ポートＡの割り込みルーチンはポートＡのデータをリードし，そのビット６～１が，ポートＢから出力されたものと一致するか否かをチェックします．ポートＡのデータをリードすると ARDY が "H" になります．この ARDY は PB_7 に接続されていますので，今度はポートＢ割り込みが発生します．

ポートＢ割り込みルーチンは出力すべきデータを更

〔Prog.❷〕 Z80 PIO モード０ （図20の構成で動作する）

```
                 ;
                 ;      Z80 PIO MODE 0 PROGRAMMING
                 ;
0000             PIOAD:  EQU   0                        ;PIO PORT A DATA
0001             PIOAC:  EQU   1                        ;PIO PORT A CONTROL
0002             PIOBD:  EQU   2                        ;PIO PORT B DATA
0003             PIOBC:  EQU   3                        ;PIO PORT B CONTROL
                 ;
1000             STACK:  EQU   1000H                    ;STACK
                 ;
                 ;
```

```
0000 310010    BEGIN: LD    SP,STACK             ;SET STACK POINTER
0003 3E01             LD    A,01H                ;SET IR
0005 ED47             LD    I,A
0007 ED5E             IM    2                    ;SET MODE 2 INTERRUPT
               ;
               ;      PIO INITIALIZATION
0009 212400           LD    HL,PACMD             ;LOAD PIO PORT A COMMAND ADDRESS
000C 0603             LD    B,PACMDL-PACMD       ;LOAD PIO PORT A COMMAND LENGTH
000E 0E01             LD    C,PIOAC              ;LOAD PIO PORT A CONTROL ADDRESS
0010 EDB3             OTIR                       ;LOAD COMMAND TO PIO PORT A
0012 0605             LD    B,PBCMDL-PBCMD       ;LOAD PIO PORT B COMMAND LENGTH
0014 0E03             LD    C,PIOBC              ;LOAD PIO PORT B CONTROL ADDRESS
0016 EDB3             OTIR                       ;LOAD COMMAND TO PIO PORT B
               ;
               ;      PIO HAS BEEN INITIALIZED
0018 3E01             LD    A,01H                ;SET PBO(ASTB/="H")
001A D302             OUT   (PIOBD),A
001C 0655             LD    B,55H                ;INITIALIZE DATA
001E 78               LD    A,B                  ;LOAD IT
001F D300             OUT   (PIOAD),A            ;LOAD DATA TO PIO PORT A
0021 FB               EI                         ;ENABLE INTERRUPT
0022 18FE             JR    $
               ;
               ;      PIO PORT A COMMAND CHAIN
0024 OF        PACMD: DB    OFH                  ;MODE 0
0025 87               DB    87H                  ;ENABLE INTERRUPT
0026 00               DB    AINTV-100H           ;INTERRUPT VECTOR
0027           PACMDL:EQU   $
               ;
               ;      PIO PORT B COMMAND CHAIN
0027 CF        PBCMD: DB    OCFH                 ;MODE 3
0028 FE               DB    OFEH                 ;PBO=OUTPUT
0029 B7               DB    OB7H                 ;INTERRUPT CONTROL WORD
002A 7F               DB    07FH                 ;PBO INTERRUPTABLE
002B 02               DB    BINTV-100H           ;INTERRUPT VECTOR
002C           PBCMDL:EQU   $
               ;
               ;      PORT B INTERRUPT
002C DB02      PBINT: IN    A,(PIOBD)            ;READ PORT B
002E CB7F             BIT   7,A                  ;PB7 (ARDY) IS HIGH ?
0030 CA4600           JP    Z,ERROR              ;NO, ERROR
0033 3EFE             LD    A,OFEH               ;ACTIVATE ASTB/
0035 D302             OUT   (PIOBD),A
0037 3E01             LD    A,01H                ;INACTIVE ASTB/
0039 D302             OUT   (PIOBD),A
003B FB               EI                         ;RE-ENABLE INTERRUPT
003C ED4D             RETI
               ;
               ;       PORT A INTERRUPT
003E 78        PAINT: LD    A,B                  ;LOAD LAST DATA
003F 2F               CPL                        ;GET COMPLEMENT OF IT
0040 47               LD    B,A                  ;SAVE IT
0041 D300             OUT   (PIOAD),A            ;LOAD DATA TO PORT A
0043 FB               EI                         ;RE-ENABLE INTERRUPT
0044 ED4D             RETI
               ;
0046 76        ERROR: HLT                        ;ERROR STOP
               ;
                      ORG   0100H
0100 3E00      AINTV: DW    PAINT                ;PORT A VECTOR
0102 2C00      BINTV: DW    PBINT                ;PORT B VECTOR
               ;
0104                  END
```

〔Prog. ❸〕 Z80 PIO モード1 (図21の構成で動作する)

```
                 ;
                 ;
                 ;      Z80 PIO MODE 1 PROGRAMMING
                 ;
0000             PIOAD: EQU   0                   ;PIO PORT A DATA
0001             PIOAC: EQU   1                   ;PIO PORT A CONTROL
0002             PIOBD: EQU   2                   ;PIO PORT B DATA
0003             PIOBC: EQU   3                   ;PIO PORT B CONTROL
                 ;
1000             STACK: EQU   1000H               ;STACK
                 ;
                 ;
0000 310010      BEGIN: LD    SP,STACK            ;SET STACK POINTER
0003 3E01               LD    A,01H               ;SET IR
0005 ED47               LD    I,A
0007 ED5E               IM    2                   ;SET MODE 2 INTERRUPT
                 ;
                 ;      PIO INITIALIZATION
0009 212600             LD    HL,PACMD            ;LOAD PIO PORT A COMMAND ADDRESS
000C 0603               LD    B,PACMDL-PACMD      ;LOAD PIO PORT A COMMAND LENGTH
000E 0E01               LD    C,PIOAC             ;LOAD PIO PORT A CONTROL ADDRESS
0010 EDB3               OTIR                      ;LOAD COMMAND TO PIO PORT A
0012 0605               LD    B,PBCMDL-PBCMD      ;LOAD PIO PORT B COMMAND LENGTH
0014 0E03               LD    C,PIOBC             ;LOAD PIO PORT B CONTROL ADDRESS
0016 EDB3               OTIR                      ;LOAD COMMAND TO PIO PORT B
                 ;
                 ;      PIO HAS BEEN INITIALIZED
0018 3E01               LD    A,01H               ;SET PB0(ASTB/) HIGH
001A D302               OUT   (PIOBD),A
001C AF                 XOR   A                   ;SET INITIAL DATA
001D 47                 LD    B,A                 ;SAVE IT IN B
001E D302               OUT   (PIOBD),A           ;LOAD INITIAL DATA TO PORT B
0020 3C                 INC   A                   ;SET PB0 HIGH
0021 D302               OUT   (PIOBD),A
0023 FB                 EI                        ;ENABLE INTERRUPT
0024 18FE               JR    $                   ;LOOP ETERNALLY
                 ;
                 ;      PIO PORT A COMMAND CHAIN
0026 4F          PACMD: DB    4FH                 ;MODE 1
0027 87                 DB    87H                 ;ENABLE INTERRUPT
0028 00                 DB    AINTV-100H          ;INTERRUPT VECTOR
0029             PACMDL:EQU   $
                 ;
                 ;      PIO PORT B COMMAND CHAIN
0029 CF          PBCMD: DB    0CFH                ;MODE 3
002A 80                 DB    80H                 ;PB7=INPUT
002B B7                 DB    0B7H                ;INTERRUPT CONTROL WORD
002C 7F                 DB    07FH                ;PB7 IS INTERRUPTABLE
002D 02                 DB    BINTV-100H          ;INTERRUPT VECTOR
002E             PBCMDL:EQU   $
                 ;
                 ;      PORT A INTERRUPT
002E DB00        PAINT: IN    A,(PIOAD)           ;READ PORT A
0030 A8                 XOR   B                   ;XOR WITH EXPECTED DATA
0031 E67E               AND   7EH                 ;STRIP-OFF BIT 0 AND 7
0033 CA4C00             JP    Z,ERROR             ;DATA OK ? NO, ERROR
0036 FB                 EI                        ;RE-ENABLE INTERRUPT
0037 ED4D               RETI
                 ;
```

```
                 ;        PORT B INTERRUPT
0039 DB02        PBINT:   IN    A,(PIOBD)        ;READ PORT B
003B CB7F                 BIT   7,A              ;ARDY IS SET ?
003D CA4C00               JP    Z,ERROR          ;NO, ERROR
0040 04                   INC   B                ;BUMP DATA
0041 78                   LD    A,B              ;LOAD IT
0042 E6FE                 AND   0FEH             ;RESET BIT 0 FOR ASTB/="L"
0044 D302                 OUT   (PIOBD),A        ;LOAD DATA
0046 3C                   INC   A                ;SET BIT 0 FOR ASTB/="H"
0047 D302                 OUT   (PIOBD),A        ;INACTIVE ASTB/
0049 FB                   EI                     ;RE-ENABLE INTERRUPT
004A ED4D                 RETI
                 ;
004C 76          ERROR:   HLT                    ;ERROR STOP
                 ;
                          ORG   0100H
0100 2E00        AINTV:   DW    PAINT            ;PORT A VECTOR
0102 3900        BINTV:   DW    PBINT            ;PORT B VECTOR
                 ;
0104                      END
```

新したのち，それをポートBに出力し，再度PB_0を通じて$\overline{ASTB}$を出力します．この過程を永遠に続けることにします．以上の動作を実行するプログラムを**Prog. ❸**に示します．

・モード2（双方向モード）

図22にZ80 PIOをモード2で動作させるための構成を示します．ポートAをモード2に設定し，ポートBはモード3に設定します．そしてポートBのPB_7は入力とし，ARDYと接続します．PB_0～PB_2は出力とし，PB_0は$\overline{ASTB}$と接続，PB_1は$\overline{BSTB}$と接続します．さらにPB_2はLS374の$\overline{OE}$に接続します．

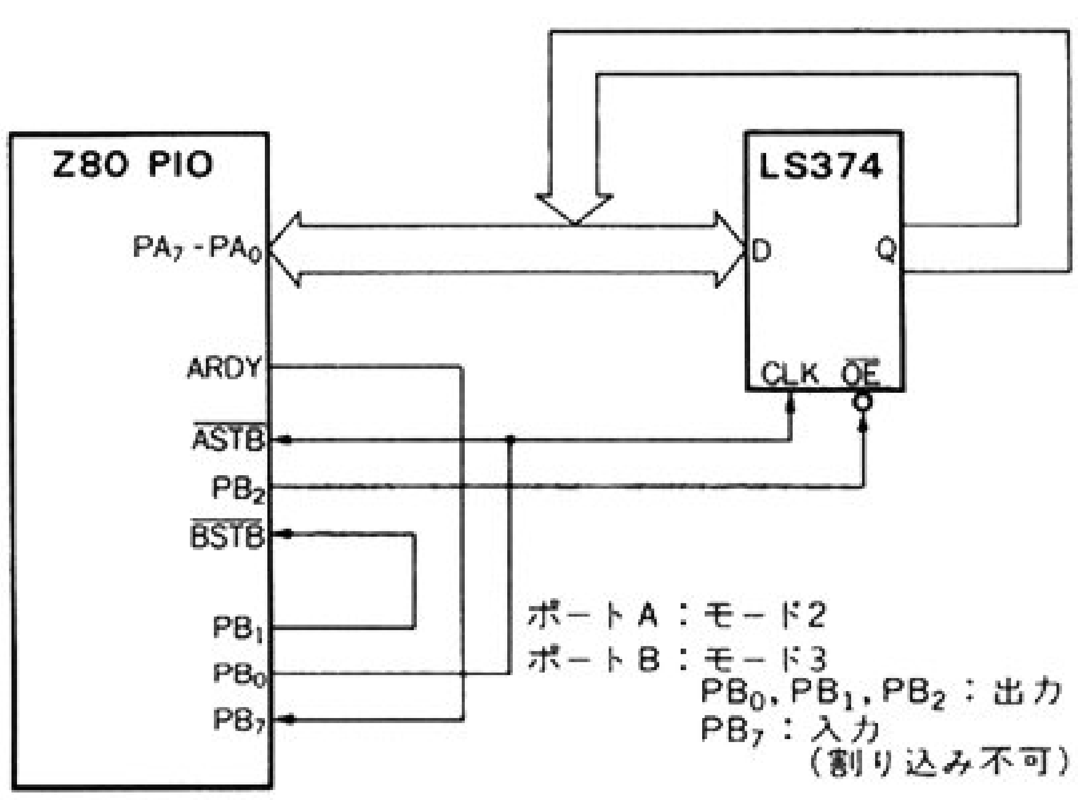
前述したように，ポートAをモード2に設定したときには，ポートAがA／B両ポートの割り込みを使いますので，ポートBの入力には，割り込みマスクをセットする必要があります．

さてプログラムはまずポートAに対してデータをライトします．これによりARDYが"H"になりますので，ポールド・モードにより，PB_7を監視しこれを感知します．次にプログラムはPB_0を通じて$\overline{ASTB}$を出力し，同時にポートA上のデータをLS374にラッチします．

さらにプログラムはPB_2によりLS374の出力をイネーブルし（$\overline{OE}$＝"L"），PB_1を通じて$\overline{BSTB}$を出力します．これによりLS374にラッチされていたデータはPIO内部のインプット・バッファにラッチされます．$\overline{BSTB}$が出力されたのでポートB割り込みが発生し，ポートB割り込みルーチンはデータをリードし，それが先に出力したものと一致するか否かをチェックします．

また前にPB_0を通じて$\overline{ASTB}$が出されていますので，ポートA割り込みが発生し，ポートA割り込みル

〈図22〉 Z80 PIOをモード2で動かせる構成



❶ポートAに対してデータをライトすると，ARDYが"H"になる
❷ポールド・モードでPB_7（ARDY）を監視し，PB_7が"H"であることを検出すると，PB_0を通じて$\overline{ASTB}$を出力し，データをLS374にラッチさせる
❸次にPB_2を"H"にしてラッチしたデータをPA_7～PA_0上に乗せたのち，PB_1により$\overline{BSTB}$を出力する
❹❷の$\overline{ASTB}$によるポートA割り込みにより，ポートAに対して次のデータをライトし，❸の$\overline{BSTB}$によるポートB割り込みにより，データを読み込みデータをチェックする

〔Prog. ❹〕 Z80 PIO モード 2 （図22の構成で動作する）

```
                ;
                ;       Z80 PIO MODE 2 PROGRAMMING
                ;
0000            PIOAD:  EQU   0                   ;PIO PORT A DATA
0001            PIOAC:  EQU   1                   ;PIO PORT A CONTROL
0002            PIOBD:  EQU   2                   ;PIO PORT B DATA
0003            PIOBC:  EQU   3                   ;PIO PORT B CONTROL
                ;
1000            STACK:  EQU   1000H               ;STACK
                ;
                ;
0000 310010     BEGIN:  LD    SP,STACK            ;SET STACK POINTER
0003 3E01               LD    A,01H               ;SET IR
0005 ED47               LD    I,A
0007 ED5E               IM    2                   ;SET MODE 2 INTERRUPT
                ;
                ;       PIO INITIALIZATION
0009 214600             LD    HL,PACMD            ;LOAD PIO PORT A COMMAND ADDRESS
000C 0603               LD    B,PACMDL-PACMD      ;LOAD PIO PORT A COMMAND LENGTH
000E 0E01               LD    C,PIOAC             ;LOAD PIO PORT A CONTROL ADDRESS
0010 EDB3               OTIR                      ;LOAD COMMAND TO PIO PORT A
0012 0605               LD    B,PBCMDL-PBCMD      ;LOAD PIO PORT B COMMAND LENGTH
0014 0E03               LD    C,PIOBC             ;LOAD PIO PORT B CONTROL ADDRESS
0016 EDB3               OTIR                      ;LOAD COMMAND TO PIO PORT B
                ;
                ;       PIO HAS BEEN INITIALIZED
0018 3E07               LD    A,07H               ;ASTB/="H", BSTB/="H", OE/="H"
001A D302               OUT   (PIOBD),A
001C 010000             LD    BC,0                ;SET-UP DATA HOLDER
                ;       B=DATA TO BE OUTPUT
                ;       C=DATA TO BE INPUT
001F 50                 LD    D,B                 ;CLEAR PORT B INTERRUPT RECEIVED
                ;       D=PORT B INTERRUPT RECEIVED
0020 78                 LD    A,B                 ;LOAD INITIAL DATA
0021 D300               OUT   (PIOAD),A
0023 FB                 EI                        ;ENABLE INTERRUPT
                ;
                ;       CHECK ARDY/
0024 DB02       CKARDY: IN    A,(PIOBD)           ;READ PORT B
0026 CB7F               BIT   7,A                 ;PB7 (ARDY) IS SET ?
0028 28FA               JR    Z,CKARDY            ;NO, CKARDY
                ;
                ;       ISSUE ASTB/
002A 3EFE               LD    A,0FEH              ;ISSUE BSTB/ (PB0)
002C D302               OUT   (PIOBD),A
002E 3C                 INC   A                   ;INACTIVATE BSTB/
002F D302               OUT   (PIOBD),A
                ;
                ;       ISSUE BSTB/
0031 F3                 DI                        ;DISABLE INTERRUPT
0032 3EF9               LD    A,0F9H              ;ISSUE BSTB/ & LS374'S OE/
0034 D302               OUT   (PIOBD),A
0036 3EFB               LD    A,0FBH              ;INACTIVATE BSTB/
0038 D302               OUT   (PIOBD),A
003A 3EFF               LD    A,0FFH              ;INACTIVATE LS374'S OE/
003C D302               OUT   (PIOBD),A
003E FB                 EI                        ;RE-ENABLE INTERRUPT
                ;
```

```
                ;       CHECK PORT B INTERRUPT
003F CB42       CKBINT:BIT  0,D                 ;PORT B INTERRUPT HAS OCCURRED ?
0041 28FC              JR   Z,CKBINT            ;NO, CKBINT
0043 15                DEC  D                   ;RESET PORT B INTERRUPT FLAG
0044 18DE              JR   CKARDY              ;GO AND CHECK NEXT ARDY
                ;
                ;       PORT A COMMAND CHAIN
0046 8F         PACMD: DB   8FH                 ;MODE 2
0047 87                DB   87H                 ;ENABLE INTERRUPT
0048 00                DB   AINTV-100H          ;INTERRUPT VECTOR
0049            PACMDL:EQU  $
                ;
                ;       PORT B COMMAND CHAIN
0049 CF         PBCMD: DB   0CFH                ;MODE 3
004A 80                DB   80H                 ;PB7=INPUT
004B B7                DB   0B7H                ;INTERRUPT CONTROL WORD
004C FF                DB   0FFH                ;MASK ALL
004D 02                DB   BINTV-100H          ;INTERRUPT VECTOR
004E            PBCMDL:EQU  $
                ;
                ;       PORT B INTERRUPT
004E F5         PBINT: PUSH AF                  ;SAVE AF
004F DB00              IN   A,(PIOAD)           ;READ BACK DATA
0051 B9                CP   C                   ;DATA OK ?
0052 C26400            JP   NZ,ERROR            ;NO, ERROR
0055 0C                INC  C                   ;BUMP EXPECTED DATA
0056 14                INC  D                   ;SET PORT B INTERRUPT OCCURED
0057 F1                POP  AF                  ;RESTORE AF
0058 FB                EI                       ;RE-ENABLE INTERRUPT
0059 ED4D              RETI
                ;
                ;       PORT A INTERRUPT
005B F5         PAINT: PUSH AF                  ;SAVE AF
005C 04                INC  B                   ;BUMP DATA TO BE OUTPUT
005D 78                LD   A,B                 ;LOAD IT
005E D300              OUT  (PIOAD),A
0060 F1                POP  AF                  ;RESTORE AF
0061 FB                EI                       ;RE-ENABLE INTERRUPT
0062 ED4D              RETI
                ;
0064 76         ERROR: HLT                      ;ERROR STOP
                ;
                       ORG  0100H
0100 5B00       AINTV: DW   PAINT               ;PORT A VECTOR
0102 4E00       BINTV: DW   PBINT               ;PORT B VECTOR
                ;
0104                   END
```

ーチンは出力データを更新し，それをポートAに出力します．他のプログラムと同様このプログラムもエンドレスです．以上の過程を行うプログラムをProg.❹に示します．

Z80 PIOの変わった応用

Z80の周辺LSIは全部Z80 CPUのモード2割り込みをサポートする機能を持っていますが，8080A系の周辺LSIをZ80 CPUに接続し，モード2割り込みを用いるにはかなりの努力を必要とします．

たしかにASYNCのTTYとだけしかインターフェースしないシステムにはZ80 SIOはあまりにもムダであり，「オレは8251Aで十分なんだがなあ」と考える場合も少なからずでしょう．このような場合，Z80 PIOの持つビット・モードの割り込みを用いると，非常に簡単にモード2にて8251Aを使用することができます．

図23にZ80 CPUシステム中に8251Aを用い，かつモード2割り込みを使用することのできる構成を示します．図からわかるように，8251Aの割り込み要因である，RXRDY, TXRDY, TXEMPTYをZ80 PIOのポートに接続します(図23の場合PA_0～PA_2)．

PIOではこれら3本の信号線が接続されているポー

トに対してモード3を設定し，かつ対応するビットを入力と設定し，かつOR機能で割り込み可とプログラミングします．図23の例ではPA$_0$～PA$_2$のいずれか一つが"H"になると，PIOは割り込み(モード2割り込み)を発生します．つまり，PIOをインタラプト・コントローラとして用いるのです．

Z80 PIOはかなり低価格なので，このようなことは現実的であり，空いたポートは他の用途に使用することができます．

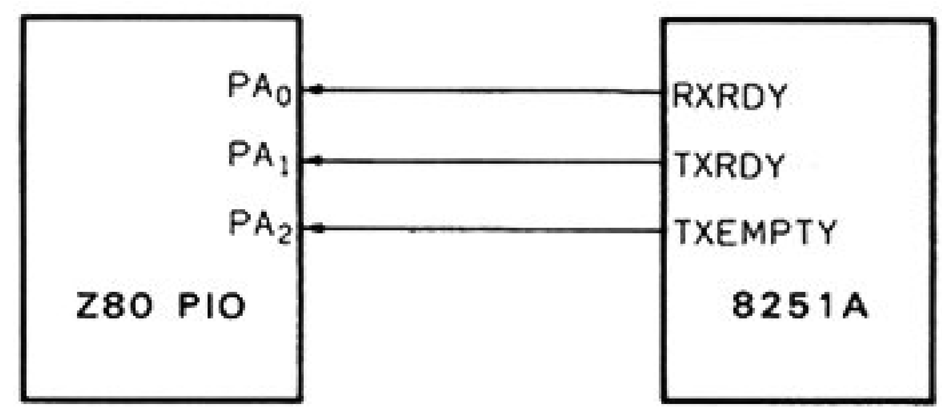
また最近のパーソナル・コンピュータでは，フロッピー・ディスクを付けるのが常識になっています．FDC(フロッピー・ディスク・コントローラ)としてシャープのLH0110を用いる場合には問題はありませんが，他のFDC，たとえばμPD765などのZ80 CPUとは無関係に作られたFDCを使用し，Z80のデイジー・チェイン式のベクトル割り込みを利用したいときにも同じような問題に直面します．

このようなときにも，FDCの割り込み信号(INT)をPIOのポート入力の1ビットに接続することにより，前記の目的を達成することができます．

Z80の周辺LSIを割り込みベクトル発生器として用いることができるのは，PIOだけに限ったことではありません．要はLSIの割り込み要因となる入力ピンが使用されずに空きとなっているか否かです．SIOの外部／ステータス割り込みの要因である，$\overline{CTS}$，$\overline{DCD}$などの入力ピンが未使用になっているときには，これらを割り込みベクトル発生器として使うことができます．

また，CTCを割り込みベクトル発生器として使う場合には，当該チャネルをカウンタ・モードに設定し，タイム・コンスタントとして1をセットしておきます．FDCなどの割り込み要求が発生すると，ダウン・カウンタの値がゼロになり，CTCは割り込みを発生します．

〈図23〉 8251AをZ80 CPUに接続し，モード2割り込みを使用する方法



3.2　Z80 CTC編

▶Z80 Counter Timer Circuit◀

マイクロコンピュータのプログラムでは，時間待ちを行うのにタイミング・ループがよく用いられますが，よく考えてみるとこれは実にもったいないプロセッサの使いかたであるといえます．逆にいうと，このようなプロセッサの使いかたができるということは非常に「幸せ」なことであり，プロセッサの処理能力をギリギリまで利用しようと，きゅうきゅうとしている世の多くのシステムからみるとうらやましい限りのことでありましょう．

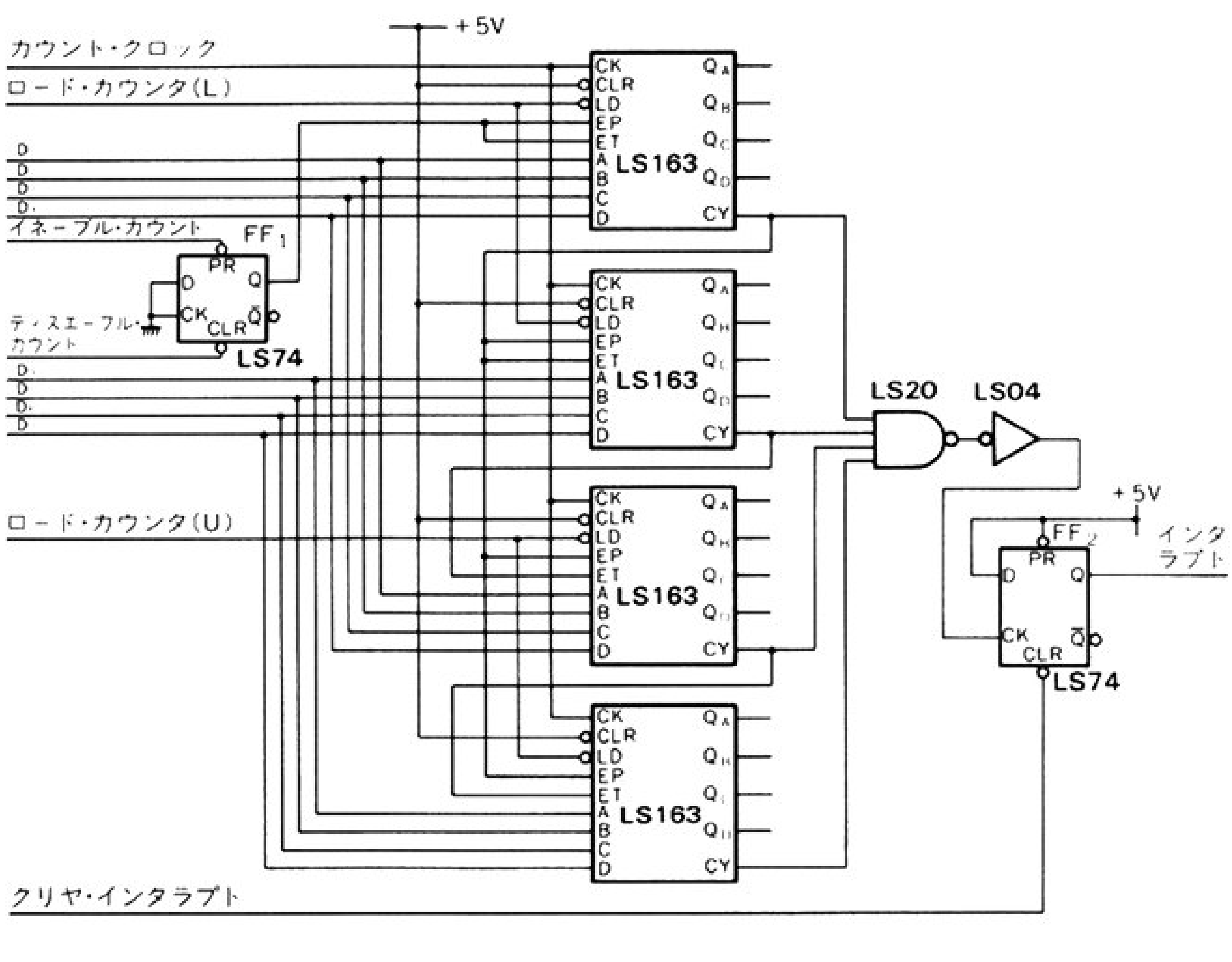
われわれのように忙しい社会人にとって，時間は貴重であり，たとえ1秒でもムダにはできません．朝などはトースタでパンを焼きながら，顔を洗ったりヒゲをそったりしています．ところが，隠居した老人の場合には時間の使用はまことに自由であり，パンを焼いている間何もしないで待っていることもできます．

これと同じように，忙しいプログラムでは1秒，いや1ミリ秒とて貴重であり，たとえ時間待ちをする場合でも，その間他の仕事をする必要があります．このような場合，ある時点から任意の時間の経過をプログラムに対して知らせてくれるハードウェアの助けを必要とします．これが本稿で述べるインターバル・タイマの働きです．

インターバル・タイマというのは，それにある時間をセットするとその時間の経過後に，通常は割り込みによりプログラムに対して時間の経過を知らせる機構のことです．つまりわれわれの周囲にある目覚し時計とそっくりの働きをします．目覚し時計の場合には，セットした時間になるまでは寝ていますが，プログラムの場合には他の仕事をしている点に違いがあります．

インターバル・タイマというとむずかしく聞こえますが，要はパラレル・ロード（プリセット）可能なダウン・カウンタのことです．つまり時間監視を要する

〈図1〉
TTLで構成した16ビット・カウンタ



ときに，必要とする時間に対応する値をカウンタにロードし，以後一定のクロックによりカウント・ダウンしていき，その値がゼロになったときに割り込み要求を発生させるロジックがあればよいのです．

このロジックをＴＴＬで作ってもたいしてむずかしくはありません．図１に74ＬＳ163（シンクロナス４ビット・バイナリ・カウンタ）を用いて作った16ビット・カウンタの例を示します．この例ではダウン・カウンタではなく，ふつうのカウンタを用いており，カウンタの値がすべて“1”になったときにインタラプト信号が出るようになっています．

図１の動作は次のとおりです．まずカウンタにセットすべき値の下位８ビットを D_0～D_7 上にセットし，ロード・カウンタ（L）を出力します．次に上位８ビットを D_0～D_7 上にセットして，ロード・カウンタ(U)を出力します．これにより16ビットのデータがカウンタにセットされます．そしてイネーブル・カウント信号を出力することにより FF_1 をセットすると，カウンタはカウント・クロックの数をカウントし始めます．

クロックの数をカウントしている間に四つのカウンタの値がすべて“1”になると，四つのカウンタのＣＹ(キャリー)が“1”になりますので，４入力ＡＮＤゲートを通じて FF_2 がセットされ，インタラプト信号が出力されます．

以上のようにカウンタ・ロジックをＴＴＬで構成することは比較的簡単ですが，現在では“カウンタ・タイマ・サーキット（例：Z80 CTC)”とか“プログラマブル・インターバル・タイマ(例：8253)”というようなＭＯＳのＬＳＩが発売されています．前者はＺ80ファミリのＩＣであり，オリジナルはザイログ社のものです．後者は8080Ａまたは8085ファミリのＩＣであり，オリジナルはインテル社のものです．

これらは28ピン（Z80 CTC)，または24ピン（8253）のＩＣであり，チップ内に８ビットまたは16ビットのカウンタを３～４個内蔵しています．そして機能の点からみても図１の例などは問題にならないくらいの豊富な機能を持っています．したがって，現在ではマイクロコンピュータ用のカウンタ回路をＴＴＬで作るのはあまり得策であるとはいいがたく，これらのＬＳＩを使用するほうが便利であり，かつ楽です．

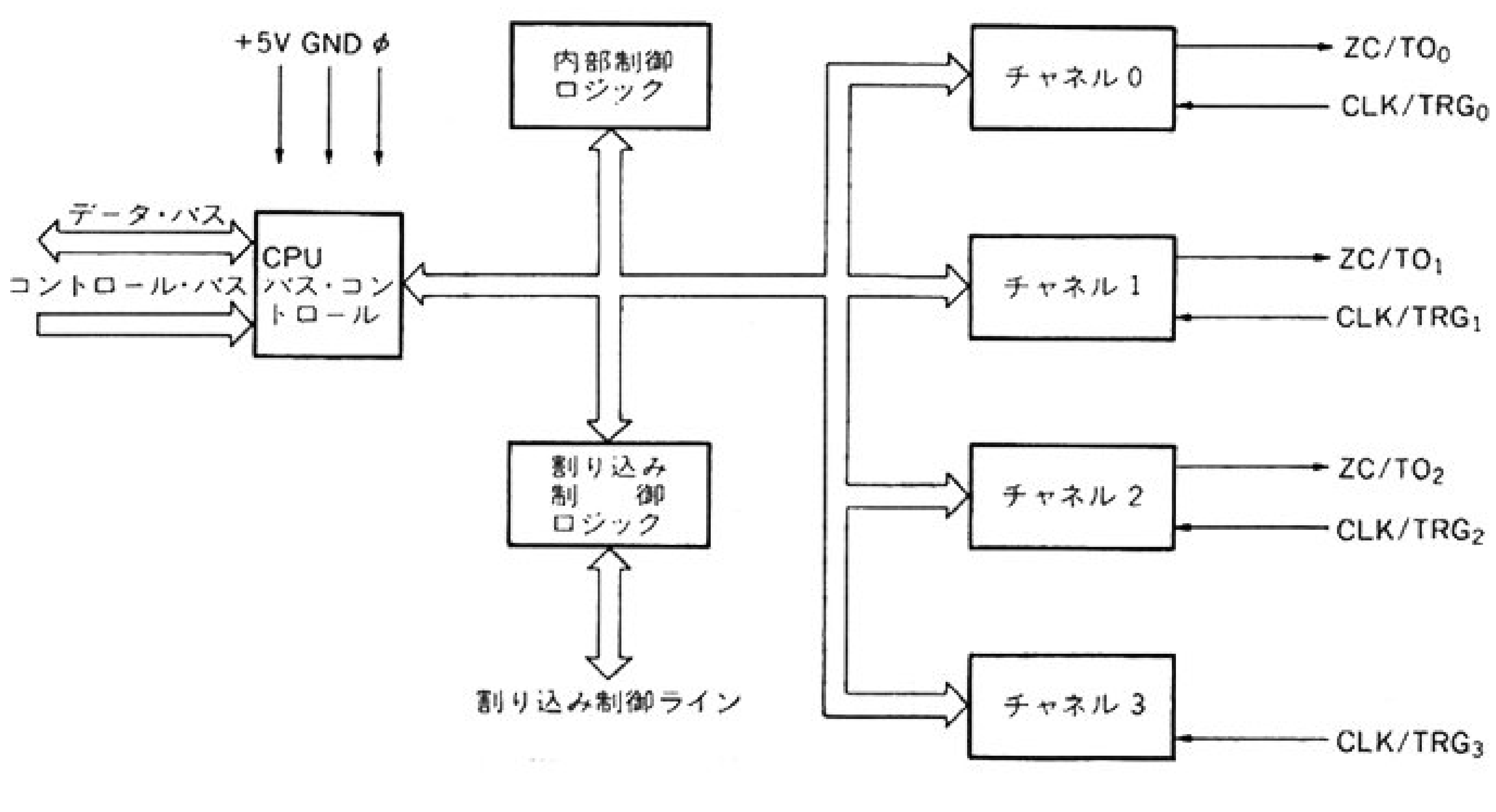
Z80 CTC

Z80 CTCも比較的早く世に現われたＺ80周辺 LSIですので，なじみの人も多いことと思います．Ｚ80 CTCの特徴を列挙すると，次のようになります．

- ５Ｖ単一電源である
- 28ピンＤＩＰである
- ４個の互いに独立した８ビット・カウンタ，または16ビット・タイマを持っている
- 各チャネルはカウンタ・モード，またはタイマ・モードで動作できる
- オート・ロード機能を持っている
- 各チャネルはシステム・クロックを16分の１，または 256 分の１するプリスケーラを持っている
- Z80 CPU のモード２割り込み機能を持っている
- デイジー・チェイン式優先割り込み制御ロジックを内蔵している
- ４チャネルのうち３チャネルがゼロ/タイム・アウト出力を持っており，これらはダーリントン・トランジスタをドライブできる
- すべての入・出力ラインは TTL コンパチブルである

Z80 CTC の構成

次に Z80 CTC の構成，および各構成要素の働きについて述べます．



〈図２〉
Z80 CTC の構成

〈図3〉 CTC各チャネルの構成

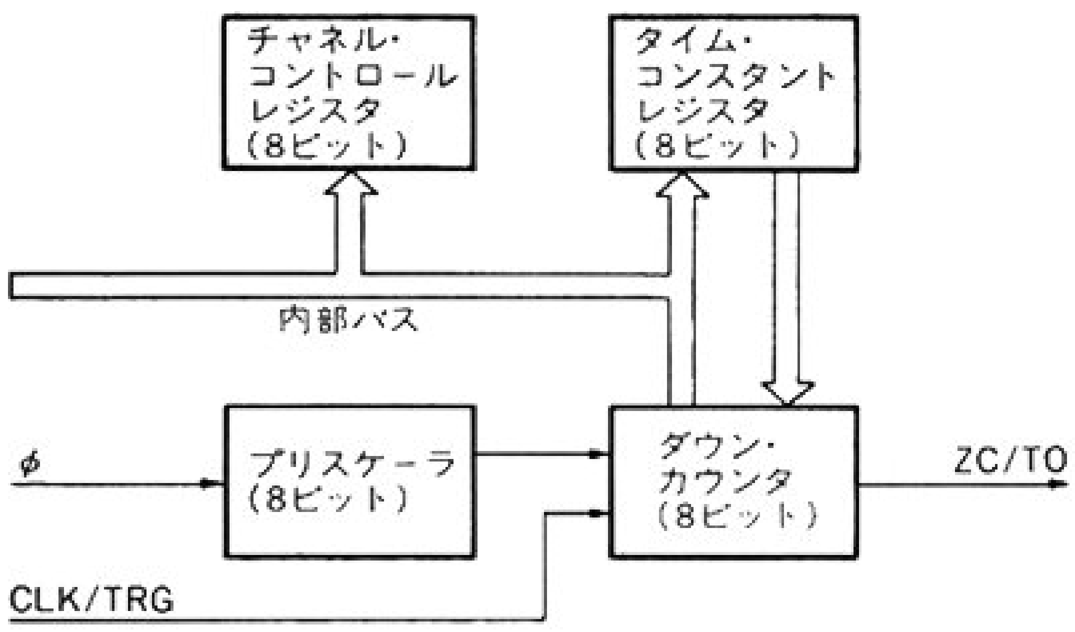


図2にZ80 CTCの構成図を示します．また図3に図2の各タイマ/カウンタ・チャネルの構成を示します．図2からわかるように，Z80 CTCはCPUバス・インターフェース，内部制御ロジック，割り込み制御ロジック，および4個のタイマ/カウンタ・チャネルからなります．各チャネルはCTCの2本の入力ピンCS_0，およびCS_1によりアドレッシングされます．

以下に，CTCのプログラミング上重要であると思われる構成要素の説明を行います．

■プリスケーラ

プリスケーラはタイマ・モードでのみ用いられ，システム・クロックを16分の1，または256分の1に分周する8ビット・カウンタです．このプリスケーラは各チャネルに一つずつ付属しており，この出力はダウン・カウンタに入力されます．プリスケーラの入力周波数の最大値は，Z80 CTCでは2.5MHz，Z80A CTCでは4MHzです．

■タイム・コンスタント・レジスタ

タイムコンスタントはTime Constantのことであり，日本語では時定数のことですが，“時定数レジスタ”ではあまりにも品がないのでそのままタイム・コンスタント・レジスタと呼びます．

このレジスタは8ビット・レジスタであり，各チャネルに一つずつあります．このレジスタにはプログラムにより1〜256の値がセットされ，CTCのチャネルがイニシャライズされたときに，当該ダウン・カウンタにその値がロードされます．またダウン・カウンタの値がゼロになると，このタイム・コンスタント・レジスタの値が自動的にダウン・カウンタにロードされます(オート・ロード)．

チャネルが動作中にこのレジスタに対して別の値がロードされたときには，その時点に実行していたダウン・カウントの終了後に新しい値がロードされます．

■ダウン・カウンタ

ダウン・カウンタもやはり8ビット・レジスタで，タイマ・モードおよびカウンタ・モード双方で用いられます．このレジスタにはCTCチャネルのイニシャライズ時，およびこのレジスタの値がゼロになったときにタイム・コンスタント・レジスタの内容がロードされます．

カウンタ・モードのときには外部クロックによりカウント・ダウンされ，タイマ・モードのときにはプリスケーラの出力によりカウント・ダウンされます．ダウン・カウンタの値がゼロになると，各チャネルのZC/TOピンの状態が1クロック期間だけに“H”になり，このときCPUに対して割り込みを要求することができます．ただしピン制限により，チャネル3にはZC/TOピンはありません．

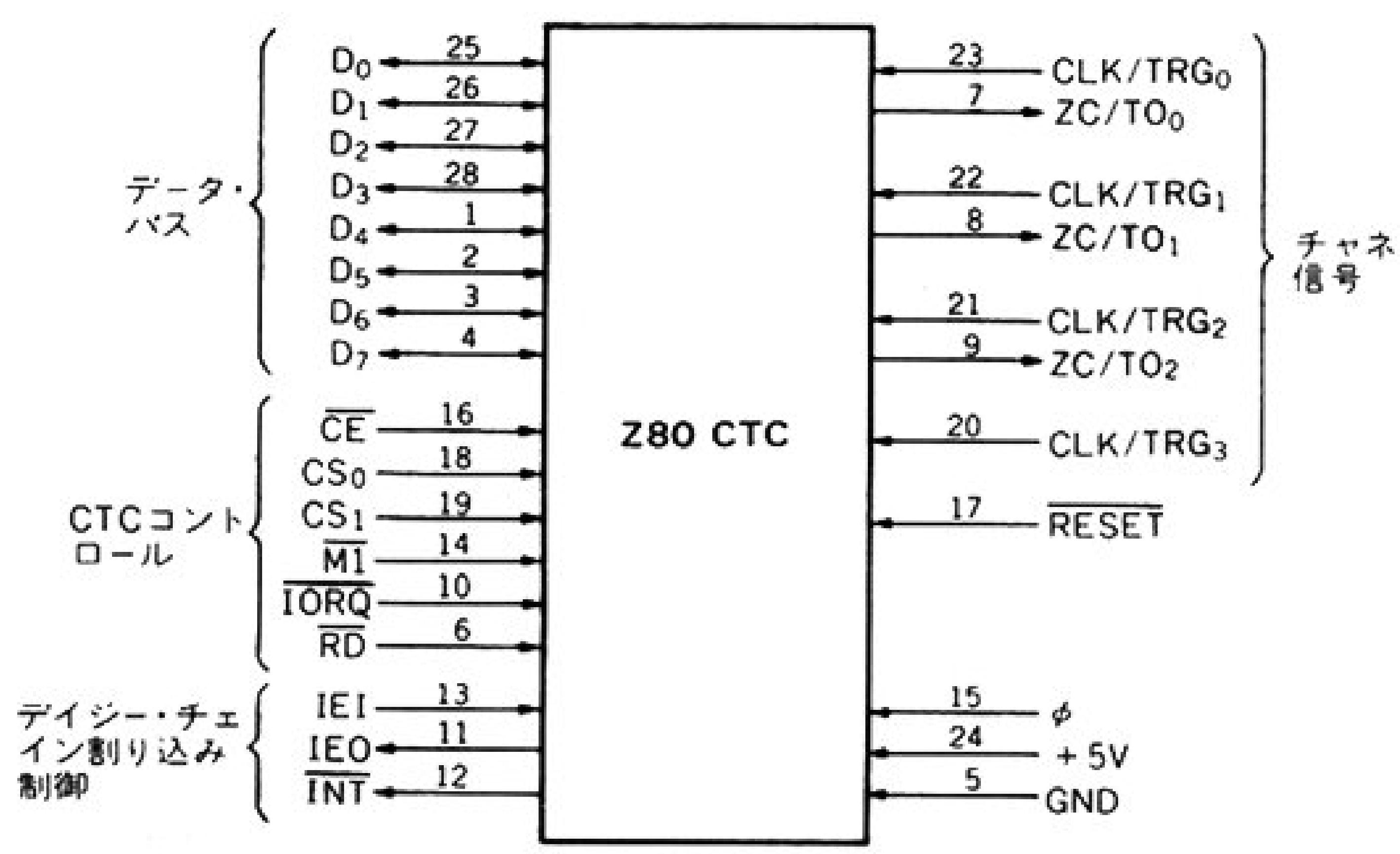
各チャネルに対して，リード命令を出すことにより，このダウン・カウンタの内容を読み取ることができます．

■割り込み制御ロジック

CTCもPIOと同様に，Z80 CPUのモード2割り込みベクトルを発生することができ，かつデイジー・チェイン式の優先割り込み制御ロジックを内蔵しています．もちろん，RETI命令の認識機能も持っています．

割り込み要求は，各チャネルのダウン・カウンタの値がゼロになったときに出されます．CTC内部にも割り込みデイジー・チェインがしかれており，優先順位はチャネル0が最高で，チャネル3が最低です．割

〈図4〉 Z80 CTCのピン接続図

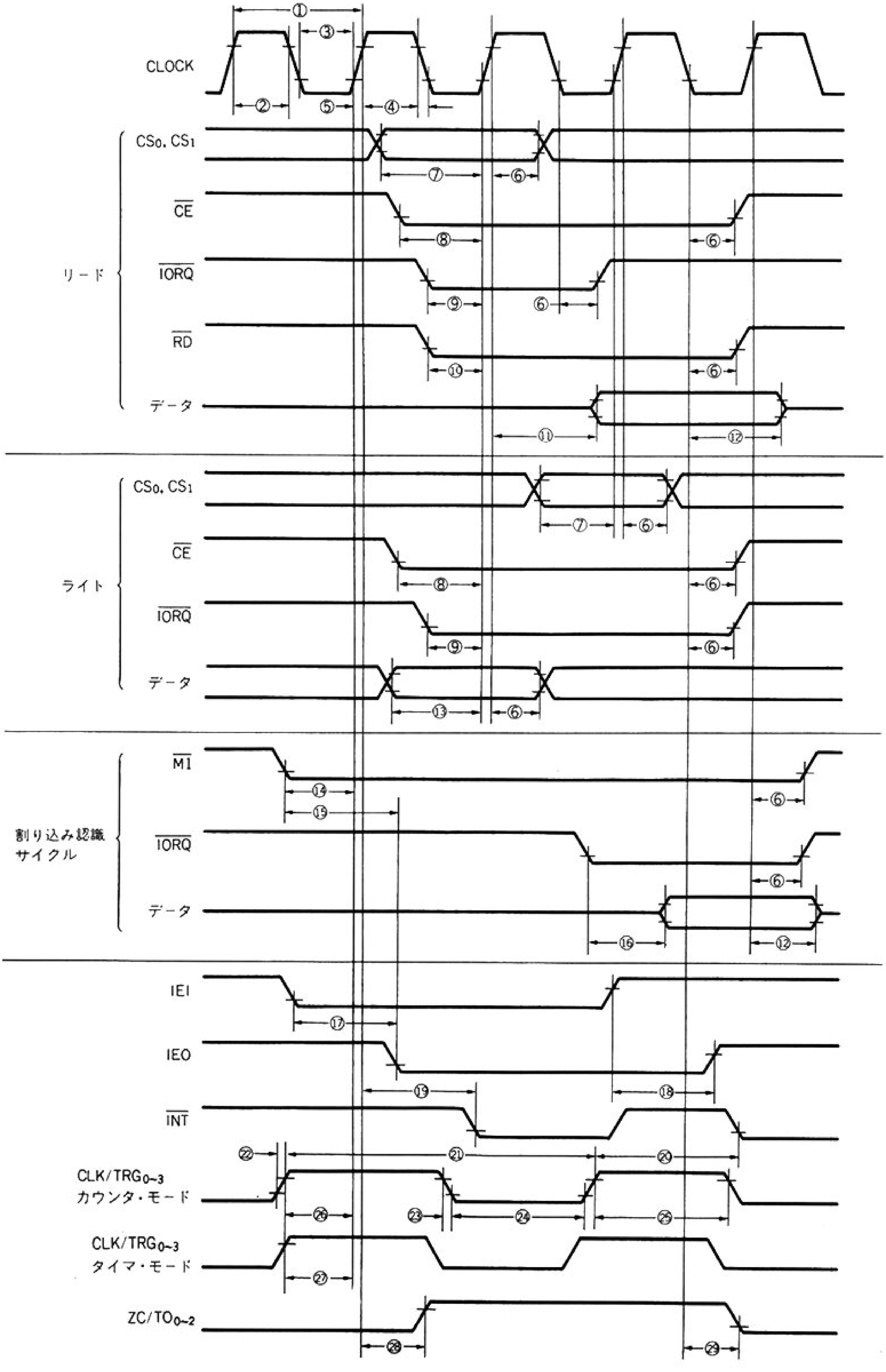


り込みベクトルは任意の一つのチャネルに対してのみロードすればよく，他のチャネルに対してはCTCが自動的に異なる値を発生してくれます．

図4にZ80 CTCのピン接置図を，図5に各ピンの機能を示します．また図6にCTCのタイミングを示し，図7に電気的特性を示します．

〈図5〉 Z80 CTCの各ピンの機能

<table>
<tr><th>ピン名</th><th>ピン番号</th><th>I/O</th><th>信号名</th><th>説明</th></tr>
<tr><td>$D_7 \sim D_0$</td><td>1～4
25～28</td><td>I/O</td><td>Z80 CPU Data Bus</td><td>Z80 CPUとZ80 CTC間のデータの授受を行うための双方向性バス．D_0がLSB．</td></tr>
<tr><td>CS0
CS1</td><td>18
19</td><td>I</td><td>Channel Select</td><td>四つのチャネルのうちの一つを選択するためのセレクト・ライン．通常はCPUのアドレス・バスA_0およびA_1を接続する．<table><tr><th></th><th>CS1</th><th>CS0</th></tr><tr><td>チャネル0</td><td>0</td><td>0</td></tr><tr><td>チャネル1</td><td>0</td><td>1</td></tr><tr><td>チャネル2</td><td>1</td><td>0</td></tr><tr><td>チャネル3</td><td>1</td><td>1</td></tr></table></td></tr>
<tr><td>$\overline{CE}$</td><td>16</td><td>I</td><td>Chip Enable</td><td>この信号が"L"であるとPIOはイネーブルされる．PIOに対してデータの読み書きを行うときには，この信号を"L"にする必要がある．通常はアドレス・デコーダの出力を接続する．</td></tr>
<tr><td>φ</td><td>15</td><td>I</td><td>System Clock</td><td>Z80 CPUに対するクロックと同じものを用いる．</td></tr>
<tr><td>$\overline{M1}$</td><td>14</td><td>I</td><td>Machine Cycle One</td><td>$\overline{M1}$および$\overline{IORQ}$が"L"のときに，CTCは割り込みベクトルをCPUに対して送る．</td></tr>
<tr><td>$\overline{IORQ}$</td><td>10</td><td>I</td><td>I/O Request</td><td>$\overline{IORQ}$は$\overline{CE}$および$\overline{RD}$と関連して，CPUとCTC間のデータのやりとりを制御する．ライト・サイクルでは$\overline{IORQ}$と$\overline{CE}$が"L"で，$\overline{RD}$が"H"である．$\overline{WR}$はCTCには入力されず，$\overline{RD}$が"H"であることを$\overline{WR}$であると解釈する．リード・サイクルでは$\overline{IORQ}$，$\overline{CE}$および$\overline{RD}$がともに"L"である．
$\overline{IORQ}$および$\overline{M1}$が"L"のときは，割り込み認識シーケンスとなる．</td></tr>
<tr><td>$\overline{RD}$</td><td>6</td><td>I</td><td>READ</td><td>この信号は$\overline{IORQ}$および$\overline{CE}$と関連して，Z80 CPUとCTC間のデータのやりとりを制御する．</td></tr>
<tr><td>IEI</td><td>13</td><td>I</td><td>Interrupt Enable In</td><td>これはZ80の割り込みデイジー・チェイン用の信号であり，この信号が"H"である場合にのみ，CTCは割り込みベクトルを送出することができ，かつRETIシーケンスを認識することができる．</td></tr>
<tr><td>IEO</td><td>11</td><td>O</td><td>Interrupt Enable Out</td><td>これはIEIと関連して，Z80の割り込みデイジー・チェインを制御する信号であり，IEIが"L"であればIEOは無条件に"L"となる．またIEIが"H"で当該CTCに割り込み要求があるときは，IEOは"L"となる．</td></tr>
<tr><td>$\overline{INT}$</td><td>12</td><td>I</td><td>Interrupt Request</td><td>CTCの各チャネルが割り込み可の状態であり，かつダウン・カウンタの値がゼロになったときに，この信号が"L"となりCPUに対して割り込み要求を行う．オープン・ドレイン型の出力である．</td></tr>
<tr><td>$\overline{RESET}$</td><td>17</td><td>I</td><td>Reset</td><td>この信号が"L"になると，全チャネルの動作が停止し，チャネル・コントロール・レジスタの割り込み可ビットはクリヤされる．ZC/TOおよび$\overline{INT}$出力はインアクティブ状態となり，IEOはIEIの状態に追随し，データ・バス・ドライバはハイ・インピーダンスになる．
この信号は最低3クロック期間"L"にする必要がある．</td></tr>
<tr><td>CLK/TRG0
CLK/TRG1
CLK/TRG2
CLK/TRG3</td><td>23
22
21
20</td><td>I</td><td>CLOCK/TRIGGER</td><td>四つのチャネルおのおのに対して一つずつCLK/TRGピンがある．カウンタ・モードではこのピンに入力されたパルスのエッジ（立ち上がり，または立ち下がり）により，ダウン・カウンタはトリガされる．タイマ・モードでは，このピンに入力されたパルスのエッジ（立ち上がり，または立ち下がり）によりタイマ動作を起動させることができる．</td></tr>
<tr><td>ZC/TO0
ZC/TO1
ZC/TO2</td><td>7
8
9</td><td>O</td><td>ZERO COUNT/
TIME OUT</td><td>このピンはチャネル0～2に対してのみ用意され，チャネル3にはない．カウンタ・モードおよびタイマ・モードの両者において，ダウン・カウンタの値がゼロになったときに，この信号は"H"となる．</td></tr>
</table>


CLOCK
CS0, CS1
CE
IORQ
RD
データ
リード
ライト
M1
割り込み認識
サイクル
IEI
IEO
INT
CLK/TRG0~3
カウンタ・モード
CLK/TRG0~3
タイマ・モード
ZC/TO0~2

CTCのタイミング

番号	記号	パラメータ	Z80-CTC 最小値	Z80-CTC 最大値	Z80A-CTC 最小値	Z80A-CTC 最大値	Z80B-CTC 最小値	Z80B-CTC 最大値	注
1	T_{cC}	クロックのサイクル・タイム	400	(1)	250	(1)	165	(1)	
2	T_{wCH}	クロックの幅("H")	170	2000	150	2000	65	2000	
3	T_{wCL}	クロックの幅("L")	170	2000	105	2000	65	2000	
4	T_{fC}	クロックの立ち下がり時間		30		30		20	
5	T_{rC}	クロックの立ち上がり時間		30		30		20	
6	T_h	ホールド・タイム	0		0		0		
7	$T_{sCS(C)}$	クロックの立ち上がりに対するCSのセットアップ・タイム	250		160		100		
8	$T_{sCE(C)}$	クロックの立ち上がりに対する$\overline{CE}$のセットアップ・タイム	200		150		100		
9	$T_{sIO(C)}$	クロックの立ち上がりに対する$\overline{IORQ}$のセットアップ・タイム	250		115		70		
10	$T_{sRD(C)}$	クロックの立ち上がりに対する$\overline{RD}$のセットアップ・タイム	240		115		70		
11	$T_{dC(DO)}$	クロックの立ち上がりに対するデータの遅延時間		240		200		130	(2)
12	$T_{dC(DOz)}$	クロックの立ち下がりに対するデータのフロート遅延時間		230		110		90	
13	$T_{sDI(C)}$	クロックの立ち上がりに対するデータ入力のセットアップ・タイム	60		50		40		
14	$T_{sM1(C)}$	クロックの立ち上がりに対する$\overline{M1}$のセットアップ・タイム	210		90		70		
15	$T_{dM1(IEO)}$	$\overline{M1}$の立ち下がりに対する$\overline{IEO}$の遅延時間		300		190		130	(3)
16	$T_{dIO(DOI)}$	$\overline{IORQ}$の立ち下がりに対するデータ出力の遅延時間(INTAサイクル)		340		160		110	(2)
17	$T_{dIEI(IEOf)}$	IEIの立ち下がりに対するIEOの遅延時間		190		130		100	(3)
18	$T_{dIEI(IEOr)}$	IEIの立ち上がりに対するIEOの遅延時間		220		160		110	(3)
19	$T_{dC(INT)}$	クロックの立ち上がりに対する$\overline{INT}$の遅延時間		T_{cC} +200		T_{cC} +140		T_{cC} +120	(4)
20	$T_{dCLK(INT)}$	CLK/TRGの立ち上がりに対する$\overline{INT}$の遅延時間 (7) (8)		⑲+㉖ ①+⑲+㉖		⑲+㉖ ①+⑲+㉖		⑲+㉖ ①+⑲+㉖	(5)
21	T_{cCTR}	CLK/TRGのサイクル・タイム	$2 \times T_{cC}$		$2 \times T_{cC}$		$2 \times T_{cC}$		
22	T_{rCTR}	CLK/TRGの立ち上がり時間		50		50		40	
23	T_{fCTR}	CLK/TRGの立ち上がり時間		50		50		40	
24	T_{wCTRL}	CLK/TRGのパルス幅("L")	200		200		120		
25	T_{wCTRH}	CLK/TRGのパルス幅("H")	200		200		120		
26	$T_{sCTR(Cs)}$	即カウントに入るための，クロックに対するCLK/TRGのセットアップ・タイム	300		210		150		(5)
27	$T_{sCTR(Ct)}$	次のクロックでプリスケーラをイネーブルするためのクロックに対するCLK/TRGのセットアップ・タイム	210		210		150		(4)
28	$T_{dC(ZC/TOr)}$	クロックの立ち上がりに対するZC/TOの遅延時間		260		190		140	
29	$T_{dC(ZC/TOf)}$	クロックの立ち下がりに対するZC/TOの遅延時間		190		190		140	

【注】

(単位：ns)

(1) $T_{cC} = T_{wCH} + T_{wCL} + T_{rC} + T_{fC}$

(2)負荷容量が50pF増えるごとに，10ns増加する．

負荷容量最大値 { データ　　　：200PF / コントロール：100pF }

(3)負荷容量が10pF増えるごとに(最高100pF)2ns増加する．

(4)タイマ・モード

(5)カウンタ・モード

(6)$\overline{RESET}$信号は最低3クロック期間"L"にする必要がある．

(7)$t_{sCTR(Cs)}$，$t_{sCTR(Ct)}$を満足しているとき

(8)$t_{sCTR(Cs)}$，$t_{sCTR(Ct)}$を満足していないとき

<図7>
Z80 CTCの電気的特性

記号	パラメータ	最小値	最大値	単位	テスト条件
V_{ILC}	クロック入力の"L"電圧	−0.3	+0.45	V	
V_{IHC}	クロック入力の"H"電圧	$V_{CC}-0.6$	$V_{CC}+0.3$	V	
V_{IL}	入力"L"電圧	−0.3	+0.8	V	
V_{IH}	入力"H"電圧	+2.0	V_{CC}	V	
V_{OL}	出力"L"電圧		+0.4	V	$I_{OL}=2mA$
V_{OH}	出力"H"電圧	+2.4		V	$I_{OH}=-250\mu A$
I_{CC}	電源電流		20	mA	
I_{LI}	入力漏れ電流		±10	μA	$V_{IN}=0\sim V_{CC}$
I_{LO}	フロート時の出力漏れ電流		±10	μA	$V_{OUT}=0.4\sim V_{CC}$
I_{OHD}	ダーリントン駆動電流	−1.5		mA	$V_{OH}=1.5V$ $R_{EXT}=390\Omega$

記号	パラメータ	最大値	単位
C_{CLK}	クロック入力容量	20	pF
C_{IN}	入力容量	5	pF
C_{OUT}	出力容量	10	pF

Z80 CTCの動作モード

Z80 CTCには，カウンタ・モードとタイマ・モードという二つの動作モードがありますが，CTCをプログラミングするに当たって，これら二つの動作モードを理解することは重要です．

■カウンタ・モード

このモードでは，CTCはCLK/TRG入力の信号のエッジをカウントします．エッジの極性（立ち上がり，または立ち下がり）はプログラムすることができます．CTCは有効エッジを検出すると，次のシステム・クロックφの立ち上がり時点にダウン・カウンタを1だけ減じます．外部クロックのエッジとφとの間にはセットアップ時間は設けられていませんが，外部クロックとφとのタイミングによっては，1φ期間だけカウント・ダウンが遅れることがあります．このモードではプリスケーラは用いられませんので，カウント可能な外部クロックの数は1～256までです．図8にカウンタ・モードのタイミングを示します．

■タイマ・モード

このモードではCTCはプリスケーラの出力をカウントします．したがってカウント可能な範囲は，プリスケーラが16分の1か256分の1にセットされているかにより，それぞれ16φ～4096φ(16φおき)，および256φ～65536φ(256φおき)です．

このモードでは，タイマをスタートさせるのに二通りの方法があります．その一つは，チャネルが初期化されるのと同時にタイマの動作が開始し，他方はCLK/TRG入力にトリガ信号が印加されるとタイマが動作し始めます．このトリガ信号の有効極性もプログラム可能です．

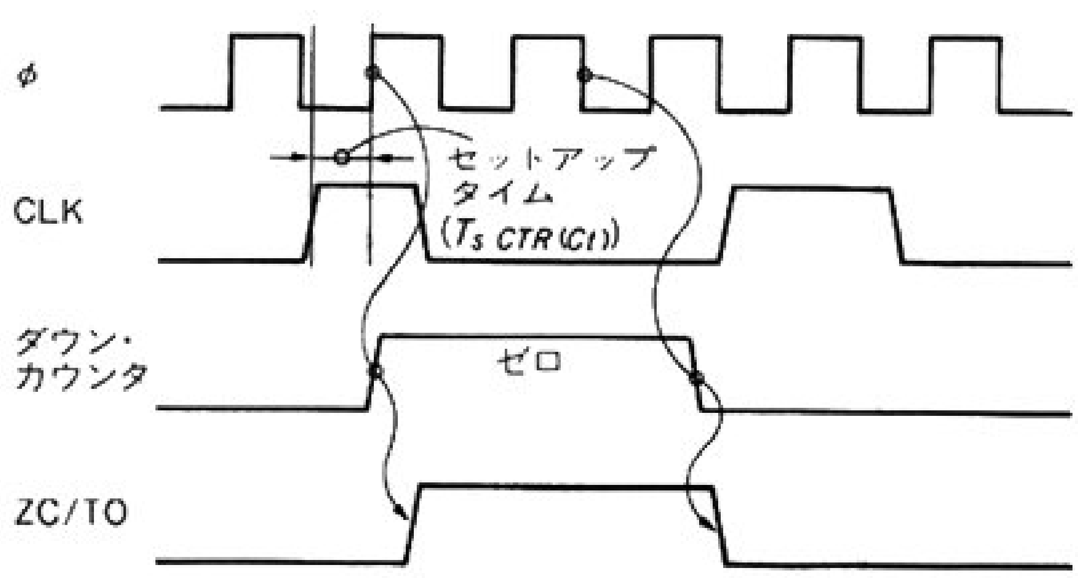
前者では，当該チャネルのタイム・コンスタント・レジスタに対するI／Oライト命令の直後のクロックφの立ち上がり時点から，タイマは動作を開始します．後者の場合には，タイム・コンスタント・レジスタに対するI／O命令が出されたのち，CLK/TRGの有効エッジが検出された直後のφの立ち上がり時にタイマ動作が開始されます．

タイム・コンスタント・レジスタに対してI／Oライト命令が出されない場合には，コントロール・ワードがライトされた後，CLK/TRGの有効エッジが検出された直後のφの立ち上がり時にタイマ動作が開始されます．図9にタイマ・モードのタイミングを示します．

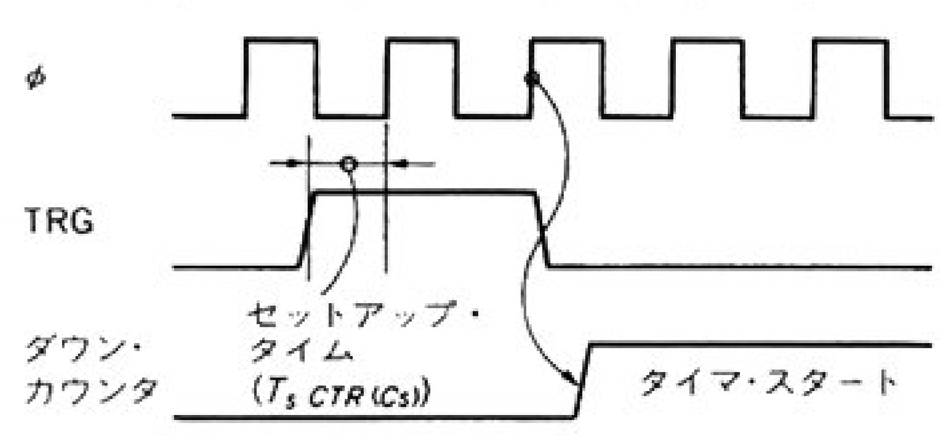
Z80 CTCのプログラミング

Z80 CTCを動作させるには使用するチャネルに対して，チャネル・コントロール・ワード，およびタイ

<図8> カウンタ・モードのタイミング

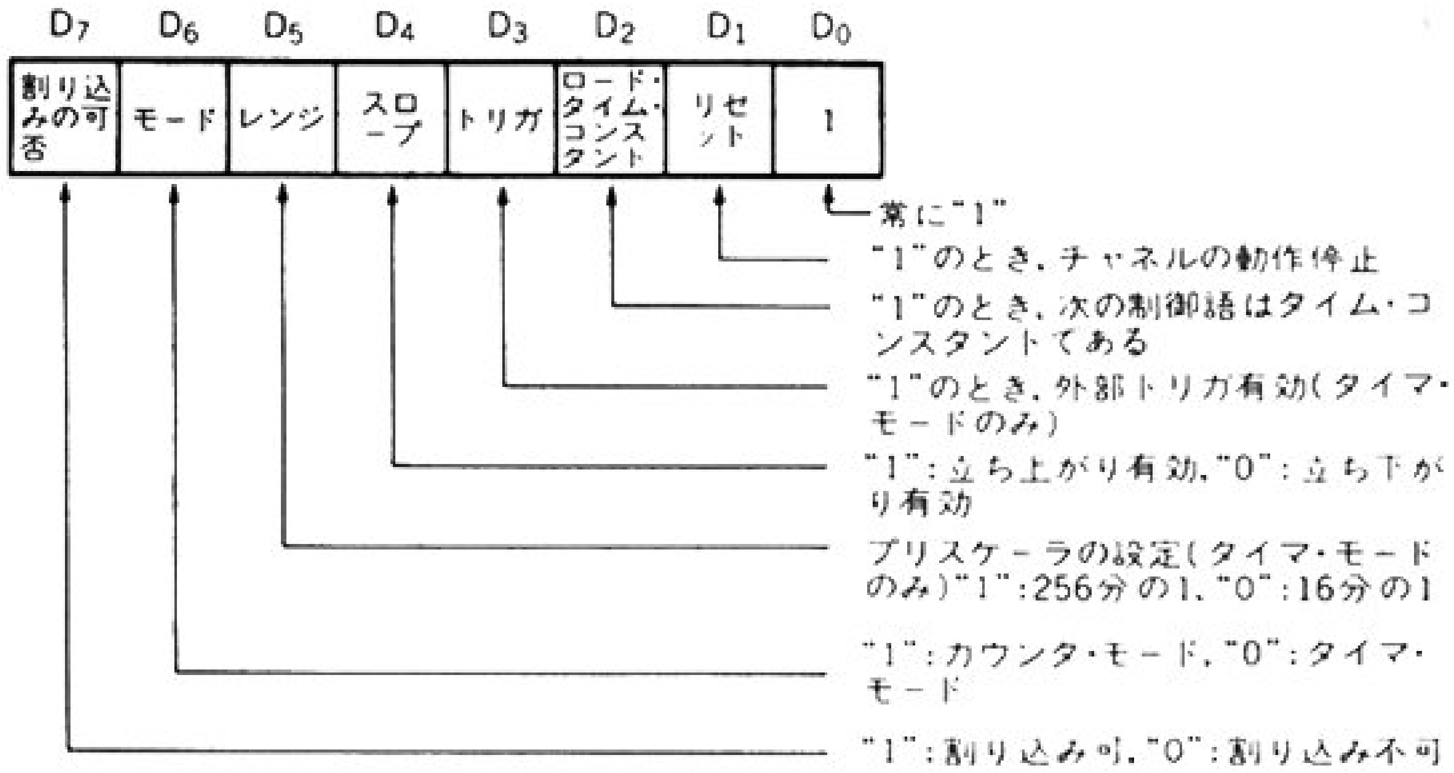


<図9> タイマ・モードのタイミング



ム・コンスタントをロードする必要があります．また割り込みを必要とする場合には，割り込みベクトルをロードする必要があります．CTC の各チャネルは入力ピン CS_0，および CS_1 により選択されます(図5参照)．通常はこれらのピンにアドレス・バスの A_0 および A_1 を接続します．

〈図10〉　CTCのチャネル・コントロール・ワード



■チャネル・コントロール・ワード

これは CTC の各チャネルの動作を規定する制御ワードであり，図10の構成を持っています．この制御ワードのビット 0 (D_0) は常に "1" であり，これにより CTC はチャネル・コントロール・ワードであるのか，割り込みベクトルであるのかの識別を行います．以下に各ビットの機能を説明します．

●ビット7(割り込みの可否)　このビットが "1" であるとダウン・カウンタの値がゼロになったときに CTC は割り込み要求を発生します．このビットを "1" にしたときは，割り込みベクトル・レジスタに割り込みベクトルをロードする必要があります．

すでに動作中の CTC チャネルに対して，このビットを "1" にしたチャネル制御ワードをロードしても，それ以前に生じたダウン・カウンタのゼロ状態が割り込みを発生することはありません．

●ビット6(動作モード)　このビットが "1" であるとカウンタ・モードが選択され，ダウン・カウンタは外部入力(CLK/TRG)のエッジによりカウント・ダウンされます．

またこのビットが "0" であるとタイマ・モードが選択され，システム・クロック ϕ によりプリスケーラはトリガされ，ダウン・カウンタはプリスケーラの出力によりカウント・ダウンされます．

●ビット5(レンジ)　このビットはタイマ・モードにおいてのみ有効であり，"1" でプリスケーラは 256 進カウンタとなり，"0" で 16 進カウンタとなります．

●ビット4(スロープ)　このビットはタイマ・モードにおけるトリガの極性，およびカウンタ・モードにおけるダウン・カウンタのトリガ極性を決めます．このビットが "1" のときは立ち上がり，"0" のときは立ち下がりとなります．

●ビット3(トリガの有無)　このビットはタイマ・モードにおいてのみ有効で，これが "1" であるとタイム・コンスタントのロード後のマシン・サイクル(CPU の) T_2 の立ち上がり時にトリガが有効となります．

トリガのセットアップ・タイムが満足されている場合には，2クロック・サイクル後にプリスケーラはカウント・ダウンされ，そうではない場合には3クロック・サイクル後になります．

また，このビットが "0" である場合には，タイム・コンスタントのロード後のマシン・サイクルの T_2 の立ち上がり時からタイマが動作を開始します．

●ビット2(ロード・タイム・コンスタント)　このビットが "1" であると，当該チャネルに対して次にタイム・コンスタントをロードすることを伝えます．

●ビット1(リセット)　このビットが "1" であると当該チャネルをリセットします．つまり，動作中のカウンタまたはタイマをストップさせます．チャネル制御ワードの他のビットには影響を与えません．

●ビット0(常に"1")　チャネル制御ワードのこのビットは常に "1" でなければなりません．このビットによりCTCはロードされたデータがチャネル制御ワードであるのか，割り込みベクトルであるのかの判別を行います．

■タイム・コンスタント・レジスタのロード

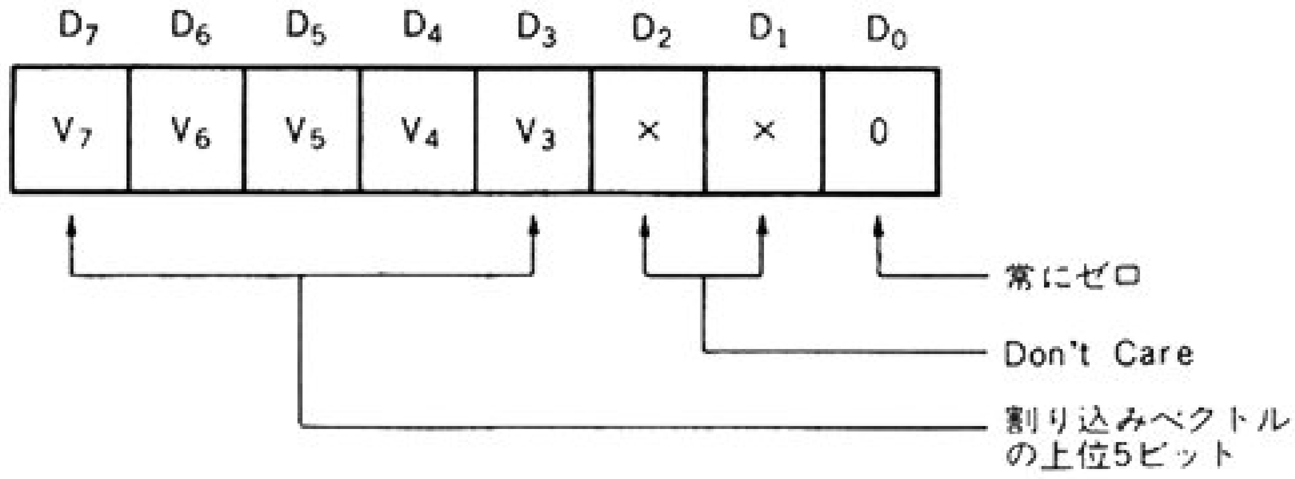
チャネル制御ワードのビット2が "1" である場合には，そのチャネルに対してロードされる次のデータはタイム・コンスタントであるとみなされます．タイム・コンスタントは8ビットであり，1～256間の任意の値をとり得ます．ただし，全ビットがゼロである場合には256であるとします．

動作中のチャネルに対してタイム・コンスタントを新たにロードしたときには，ダウン・カウンタがゼロ

〈図11〉　タイム・コンスタント・ワード(TC_7 が MSB)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
TC_7	TC_6	TC_5	TC_4	TC_3	TC_2	TC_1	TC_0

<図12>
割り込みベクトル・ワード



ビット1および2はDon't Careであるが，割り込み受け付け時にCTCにより下記の値が挿入される

	D_2	D_1
チャネル0	0	0
チャネル1	0	1
チャネル2	1	0
チャネル3	1	1

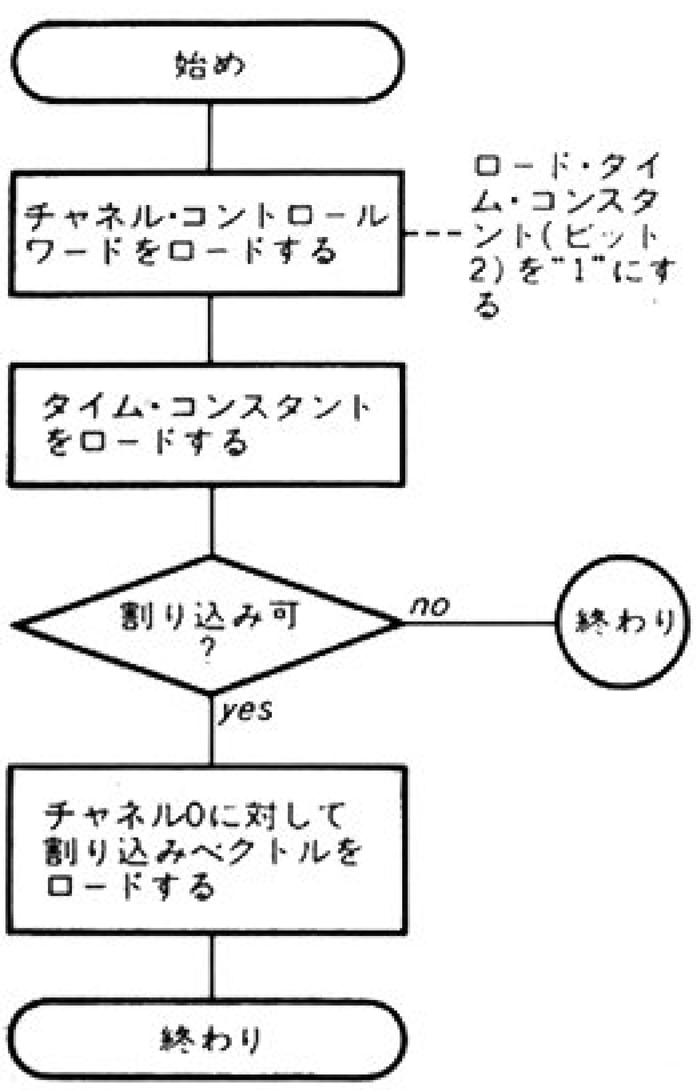
になったときにはじめて新しいタイム・コンスタントが有効になります．タイム・コンスタント・ワードの構成を図11に示します．

■割り込みベクトルのロード

図12にCTCの割り込みベクトルの構成を示します．割り込みベクトルのビット0は常にゼロであり，ビット1および2は Don't Care です．したがってプログラミングすべき値はビット3～7の5ビットです．割り込みベクトルはチャネル0に対してのみロードすればよく，他のチャネルに対する割り込みベクトルはCTCが自動的に発生します．

つまり，CTCはチャネルごとに割り込みベクトルのビット1および2の値を00，01，10，11と変化させる機能を持っています．したがって，例えばチャネル3を割り込み可としてプログラミングした場合にも，チャネル0に対して割り込みベクトルを送る必要があります．

「割り込みベクトルはチャネル0に対して送る」というのはマニュアル上のことであり，筆者の実験ではチャネル0またはチャネル2に対して割り込みベクトルを送ることができました．つまり，偶数チャネルでは割り込みベクトルを正常に送ることができました．



<図13>
CTCのプログラミング・フロー

ただし，使用したCTCはザイログ社製の Z80A CTC です．

CTCのプログラミングの項のまとめとして，図13にCTCのプログラミング・フローを示します．

Z80 CTC のプログラム例

ここでは Z80 CTC を用いたプログラム例を，三つほど示します．

■ボーレート・ジェネレータ

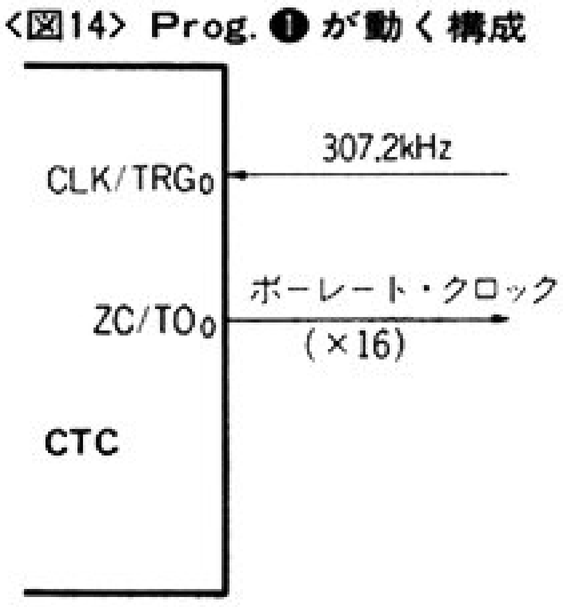
シリアル・インターフェース，とくに非同期モードのシリアル・インターフェースを用いるときには，使用する回線のボーレートに合わせたクロックを発生する，ボーレート・ジェネレータを必要とします．Prog.❶に CTC を用いたボーレート・ジェネレータを示します．また図14にこのプログラムが動作するためのハードウェア構成を示します．

このプログラムはサブルーチン形式になっており，アキュムレータの下位4ビットの値により，110～19200BPS (Bit Per Second)のクロックを発生します．なおクロックの周波数(ファクタ)はよく使われる16倍(×16)です．

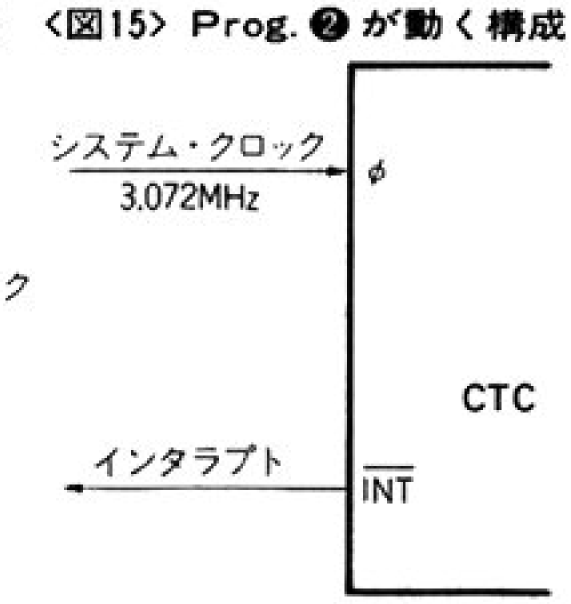
CTC チャネル0の CLK/TRG 0 には 307.2kHz のクロックが供給され，チャネル0はこのクロックをカウンタ・モードにより分周し，ZC/TO 0 にボーレート・クロックを出力します．

CTC の性質として，ZC/TOx (X＝0～2)に出力されるパルスの幅は，システム・クロック(ϕ)の周期の1.5倍程度です．Z80 SIO を用いる場合には，ZC/TOx の出力をそのまま SIO の送/受信クロックとして用い

<図14> **Prog.❶が動く構成**



<図15> **Prog.❷が動く構成**



<Prog. ❶> ボーレート・ジェネレータ

```
                    ;*****************************************************
                    ;****            BAUD RATE GENERATOR              ****
                    ;*****************************************************
                    ;       THIS ROUTINE WILL SET UP CTC CHANNEL 0 TO
                    ;       GENERATE BAUD RATE CLOCK (X16) ACCORDING TO
                    ;       CONTENT OF ACC.
                    ;       ACC     BAUD RATE SELECTOR
0000  E6 0F         BAUD:   AND     0FH             ;CLEAR GABAGE
0002  4F                    LD      C,A             ;SELECTOR TO REG. C
0003  06 00                 LD      B,0             ;ZERO TO REG. B
0005  21 0011               LD      HL,TABLE        ;LOAD BAUD RATE TABLE
                                                    ;ADDRESS
0008  09                    ADD     HL,BC           ;POINTS TO PROPER RATE
0009  3E 47                 LD      A,47H           ;LOAD CTC CONTROL WORD
                                                    ;DISABLE INTERRUPT
                                                    ;COUNTER MODE
                                                    ;LOAD TIME CONSTANT
000B  D3 08                 OUT     (CTC0),A        ;LOAD IT TO CTC 0
000D  7E                    LD      A,(HL)          ;LOAD TIME CONSTANT
000E  D3 08                 OUT     (CTC0),A        ;LOAD IT TO CTC 0
0010  C9                    RET                     ;
                    ;
                    ;       BAUD RATE TABLE
0011  AF            TABLE:  DB      175             ;0      110
0012  8F                    DB      143             ;1      134.5
0013  80                    DB      128             ;2      150
0014  60                    DB      96              ;3      200
0015  40                    DB      64              ;4      300
0016  20                    DB      32              ;5      600
0017  10                    DB      16              ;6     1200
0018  08                    DB      8               ;7     2400
0019  04                    DB      4               ;8     4800
001A  02                    DB      2               ;9     9600
001B  01                    DB      1               ;A    19200
001C  01                    DB      1               ;B    19200
001D  01                    DB      1               ;C    19200
001E  01                    DB      1               ;D    19200
001F  01                    DB      1               ;E    19200
0020  01                    DB      1               ;F    19200
                    ;
0008                CTC0    EQU     08H             ;CTC CHANNEL 0
                    ;
                            END
```

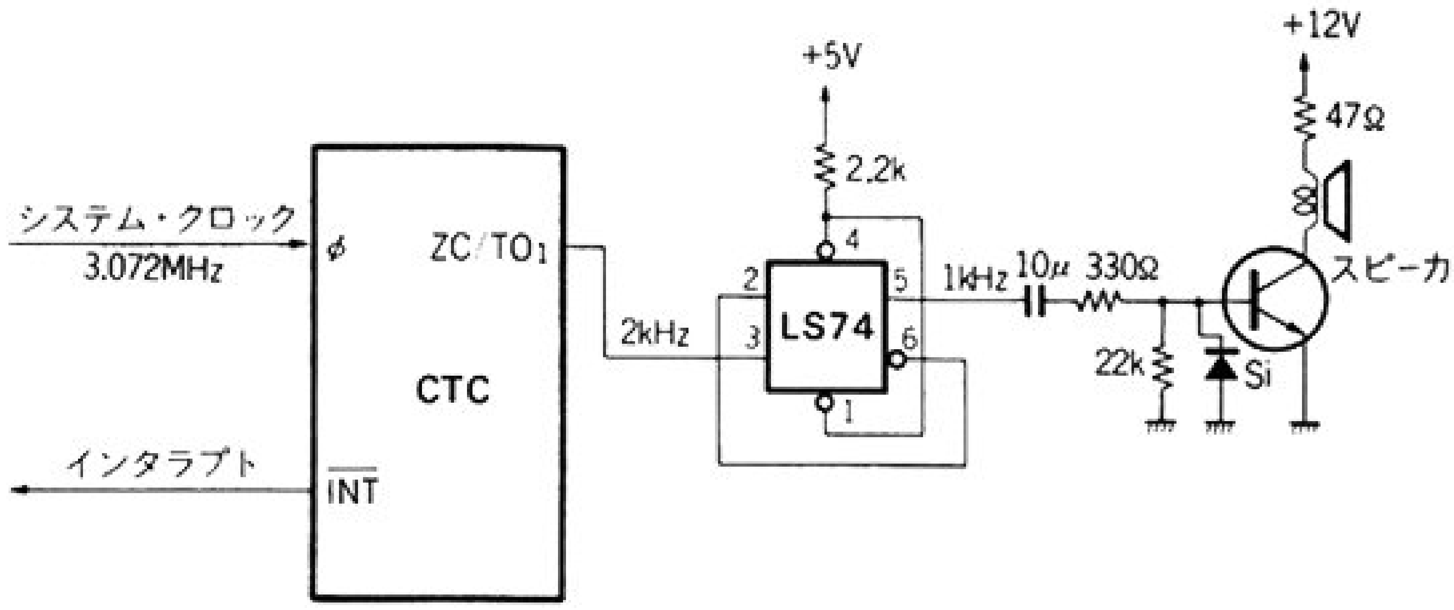
<Prog. ❷> デイ・クロック

```
                    ;****************************************************
                    ;****            DAY  CLOCK                      ****
                    ;****************************************************
                    ;       THIS ROUTINE WILL MAINTAIN DAY CLOCK USING
                    ;       CTC CHANNEL 0 INTERRUPT. CTC CHANNEL 0 IS
                    ;       PROGRAMMED TO GENERATE INTERUPT EVERY 10 MS.
0000'                       ASEG
                            ORG     2000H
2000  31 20B5       CLOCK:  LD      SP,STACK        ;SET STACK POINTER
2003  21 2020               LD      HL,VECTOR       ;LOAD INT. VECTOR ADRS
2006  7C                    LD      A,H             ;LOAD UPPER HALF OF IT
2007  ED 47                 LD      I,A             ;SET INTERRUPT REG.
2009  ED 5E                 IM      2               ;SET MODE 2 INTERRUPT
200B  3E A7                 LD      A,0A7H          ;LOAD CHANNEL CONTROL
                                                    ;ENABLE INTERRUPT
                                                    ;TIMER MODE
                                                    ;PRESCALER=256
                                                    ;LOAD TIME CONSTANT
```

```
200D    D3 08                OUT    (CTCO),A      ;LOAD CONTROL WORD
200F    3E 78                LD     A,120         ;LOAD TIME CONSTANT
2011    D3 08                OUT    (CTCO),A      ;
2013    7D                   LD     A,L           ;LOAD LOWER HALF OF INT.
                                                  ;VECTOR ADDRESS
2014    D3 08                OUT    (CTCO),A      ;
2016    FB                   EI                   ;ENABLE INTERRUPT
2017    18 FE                JR     $             ;LOOP ETERNALLY
                    ;
0008                CTCO     EQU    08H           ;CTC CHANNEL 0 ADDRESS
                    ;
                             ORG    CLOCK+20H
2020    2022        VECTOR:  DW     INT           ;CTC 0 INTERRUPT VECTOR
                    ;
                    ;        CTC CHANNEL 0 INTERRUPT ROUTINE
2022    F5          INT:     PUSH   AF            ;SAVE REGISTERS
2023    C5                   PUSH   BC            ;
2024    E5                   PUSH   HL            ;
2025    2A 2070              LD     HL,(MS)       ;LOAD MS TIMER
2028    01 000A              LD     BC,10         ;UP 10 MS
202B    09                   ADD    HL,BC         ;
202C    01 03E8              LD     BC,1000       ;LOAD 1000
202F    7C                   LD     A,H           ;MAY BE 1000 MS ?
2030    B8                   CP     B             ;
2031    20 34                JR     NZ,INT9       ;NO,INT9
2033    7D                   LD     A,L           ;1000 MS ?
2034    B9                   CP     C             ;
2035    20 30                JR     NZ,INT9       ;NO,INT9
                    ;        UP SECOND TIMER
2037    3A 2072              LD     A,(SEC)       ;LOAD SECOND TIMER
203A    3C                   INC    A             ;UP ONE SECOND
203B    32 2072              LD     (SEC),A       ;
203E    FE 3C                CP     60            ;60 SECOND ?
2040    20 22                JR     NZ,INT8       ;NO,INT8
                    ;        UP MINUTE TIMER
2042    3A 2073              LD     A,(MIN)       ;LOAD MINUTE TIMER
2045    3C                   INC    A             ;UP MINUTE TIMER
2046    32 2073              LD     (MIN),A       ;
2049    FE 3C                CP     60            ;60 MINUTES ?
204B    20 13                JR     NZ,INT7       ;NO,INT7
                    ;        UP HOUR TIMER
204D    3A 2074              LD     A,(HOR)       ;LOAD HOUR TIMER
2050    3C                   INC    A             ;UP HOUR
2051    32 2074              LD     (HOR),A       ;
2054    FE 18                CP     24            ;24 HOURS ?
2056    20 04                JR     NZ,INT6       ;NO,INT6
2058    AF                   XOR    A             ;RESET HOUR TIMER
2059    32 2074              LD     (HOR),A       ;
205C    AF          INT6:    XOR    A             ;RESET MINUTE TIMER
205D    32 2073              LD     (MIN),A       ;
2060    AF          INT7:    XOR    A             ;RESET SECOND TIMER
2061    32 2072              LD     (SEC),A       ;
2064    21 0000     INT8:    LD     HL,0          ;RESET MS TIMER
2067    22 2070     INT9:    LD     (MS),HL       ;UPDATE MS TIMER
206A    E1                   POP    HL            ;RESTORE REGISTERS
206B    C1                   POP    BC            ;
206C    F1                   POP    AF            ;
206D    FB                   EI                   ;RE-ENABLE INTERRUPT
206E    ED 4D                RETI                 ;RETURN FROM INTERRUPT
                    ;
2070    0000        MS:      DW     0             ;MS TIMER
2072    00          SEC:     DB     0             ;SECOND TIMER
2073    00          MIN:     DB     0             ;MINUTE TIMER
2074    00          HOR:     DB     0             ;HOUR TIMER
                    ;
2075                         DS     64            ;STACK AREA
20B5                STACK    EQU    $             ;STACK FRAME
                    ;
                             END
```

ることができますが，8251などを用いる場合にはデューティが約50%のクロックを必要とします．このようなときには，ZC/TO$_X$の出力周波数を2倍にしておき，フリップフロップにより，デューティ50%のパルスを得，これをボーレート・クロックとして用います．

〈図16〉 Prog.❸が動く構成



■デイ・クロック

CTCをタイマ・モードに設定し，10msごとに割り込みを発生させることにより，24時間時計を構成する例をProg.❷に示します．また図15にプログラムが動作するためのハードウェア構成を示します．

このプログラムはCTCチャネル0をタイマ・モードに設定し，256のプリスケーラを選択し，システム・クロック(3.072MHz)を(256×120)分の1することにより，10ms(100Hz)ごとに割り込みを発生させます．割り込みルーチンは割り込みが起きるたびに，ミリ・セコンド・タイマMS，秒タイマSEC，分タイマMIN，ならびに時タイマHORを更新します．

■サウンド・アラーム

マイクロコンピュータの応用では，警告音(サウンド・アラーム)がよく使われます．Prog.❸にサウンド・アラームを発生するプログラムを示します．また図16にこのプログラムが動作するためのハードウェア構成を示します．このプログラムは2種類のサウンド・アラームを出力することができます．その一つはワンショット・ビープ(One Shot Beep)であり，これは1kHzのピーという音を500ms間，1回だけ発生します．

もう一つはサイクリック・アラーム(Cyclic Alarm)であり，これは1kHzのピーという音を500ms間鳴らし，500ms間休んだ後に再びピーという音を500ms間鳴らします．つまり，ピー，ピー，ピー，…という周期的なアラーム音を発生します．

プログラムはCTCチャネル0をタイマ・モードに設定し，10msごとに割り込みを発生するようにプログラムします．ワンショット・ビープを発生したいときには，FLAGのビット0を"1"にし，サイクリック・アラームを発生したいときには，FLAGのビット1を"1"にします．

CTCチャネル0の割り込みルーチンはFLAGのビット0，または1がセットされていると，CTCチャネル1を用いて，ZC/TO1に2kHzのパルスを出力します．この2kHzのパルスはフリップフロップによりデューティ=50%の1kHzのパルスとなり，スピーカを駆動します．

ワンショット・ビープの場合には500ms間パルスを出力した後に，CTCチャネル1をリセットします．サイクリック・アラームの場合には，500ms間パルスを出力した後に，CTCチャネル1をリセットする点は，ワンショット・ビープの場合と同じですが，さらに500msが経過すると，再度CTCチャネル1から2kHzのパルスを出力します．

この500msの時間監視にCTCチャネル0が発生する10msごとの割り込みが用いられています．つまり，チャネル0の割り込み回数を50回数えることにより，500msの時間経過を測定します．サイクリック・アラームを解除するには，FLAGのビット1をゼロにリセットします．

Prog.❷およびProg.❸からおわかりのように，これらのプログラムでは経時測定を行うのに，タイミング・ループを用いたりせず，割り込みを使用しています．このため，割り込みが発生して割り込みルーチンが動く時間以外の時間を他の仕事に費すことができます．

```
                            ;************************************************************
                            ;****           SOUND  ALARM                             ****
                            ;************************************************************
0000                                ASEG
                                    ORG     2000H
2000      31 20EB           ALARM:  LD      SP,STACK            ;SET STACK POINTER
2003      21 2040                   LD      HL,VECTOR           ;LOAD INT. VECTOR ADRS
2006      7C                        LD      A,H                 ;LOAD UPPER HALF OF INT.
                                                                ;VECTOR ADDRESS
2007      ED 47                     LD      I,A                 ;LOAD INTERRUPT REG.
2009      ED 5E                     IM      2                   ;SET MODE 2
200B      3E A7                     LD      A,0A7H              ;LOAD CHANNEL CONTROL
                                                                ;ENABLE INTERRUPT
                                                                ;TIMER MODE
                                                                ;PRESCALER=256
                                                                ;LOAD TIME CONSTANT
200D      D3 08                     OUT     (CTC0),A            ;LOAD CONTROL WORD
200F      3E 78                     LD      A,120               ;LOAD TIME CONSTANT
2011      D3 08                     OUT     (CTC0),A            ;
2013      7D                        LD      A,L                 ;LOAD LOWER HALF OF
                                                                ;INTERRUPT VECTOR ADRS
2014      D3 08                     OUT     (CTC0),A            ;
2016      FB                        EI                          ;ENABLE INTERRUPT
                            ;
                            ;       SOUND ONE SHOT  BEEP
2017      21 2042                   LD      HL,FLAG             ;SET ONE SHOT BEEP
201A      36 01                     LD      (HL),01H            ;
                            ;
                            ;       SOUND CYCLIC ALARM
201C      21 2042                   LD      HL,FLAG             ;SET CYCLIC ALARM
201F      36 02                     LD      (HL),02H            ;
                            ;
                            ;       DISABLE CYCLIC ALARM
2021      21 2042                   LD      HL,FLAG             ;CLEAR SOUND
2024      36 00                     LD      (HL),00H            ;
                            ;
                            ;
0008                        CTC0    EQU     08H                 ;CTC CHANNEL 0
0009                        CTC1    EQU     CTC0+1              ;CTC CHANNEL 1
                            ;
                                    ORG     ALARM+40H
2040      2045              VECTOR: DW      INT                 ;INTERRUPT VECTOR
                            ;
2042      00                FLAG:   DB      00H                 ;SOUND FLAG
                            ;       BIT     0       ONE SHOT BEEP
                            ;       BIT     1       CYCLIC SOUND
                            ;
2043      00                FLAG0:  DB      00H                 ;CONTROL FLAG
                            ;       BIT     0       SOUND ON TIME
                            ;       BIT     1       WAITING FOR 500 MS
                            ;
2044      00                COUNT:  DB      00H                 ;100 MS COUNTER
                            ;
                            ;
                            ;       CTC CHANNEL 0 INTERRUPT
2045      F5                INT:    PUSH    AF                  ;SAVE REGISTERS
2046      E5                        PUSH    HL                  ;
2047      21 2043                   LD      HL,FLAG0            ;LOAD FLAG0 ADDRESS
204A      3A 2042                   LD      A,(FLAG)            ;LOAD FLAG
204D      E6 03                     AND     03H                 ;NEED SOUND ?
204F      28 10                     JR      Z,INT0              ;NO,INT0
2051      CB 47                     BIT     0,A                 ;SOUND ONE SHOT BEEP ?
2053      28 1E                     JR      Z,INT2              ;NO,INT2
```

```
                    ;       ONE SHOT BEEP FOR 500 MS
2055    CB 46               BIT     0,(HL)          ;SOUND ON ?
2057    20 0F               JR      NZ,INT1         ;YES,INT1
                    ;       SOUND ON
2059    CD 208A             CALL    SON             ;SOUND ON
                    ;       RETURN FROM INTERRUPT
205C    E1          INTR:   POP     HL              ;RESTORE REGISTERS
205D    F1                  POP     AF              ;
205E    FB                  EI                      ;RE-ENABLE INTERRUPT
205F    ED 4D               RETI                    ;
                    ;       MAKE SURE SOUND IS OFF
2061    CD 2096     INT0:   CALL    SOF             ;SOUND OFF
2064    36 00               LD      (HL),0          ;RESET FLAG0
2066    18 F4               JR      INTR            ;
2068    CD 20A1     INT1:   CALL    UPC             ;UP 500 MS COUNTER
206B    38 EF               JR      C,INTR          ;500 MS EXPIRED ?
                                                    ;NO,INTR
206D    AF                  XOR     A               ;RESET FLAG
206E    32 2042             LD      (FLAG),A        ;
2071    18 EE               JR      INT0            ;
                    ;       CYCLIC ALARM SOUND
2073    CB 4E       INT2:   BIT     1,(HL)          ;WAIT FOR 500 MS ?
2075    20 05               JR      NZ,INT3         ;YES,INT3
2077    CD 208A     INT20:  CALL    SON             ;SOUND ON
207A    18 E0               JR      INTR            ;
207C    CD 20A1     INT3:   CALL    UPC             ;UP 500 MS COUNTER
207F    38 DB               JR      C,INTR          ;GO TO INTR NOT IF 500
                                                    ;MS COUNTER EXPIRED
2081    CB 46               BIT     0,(HL)          ;SOUND ON TIME ?
2083    28 F2               JR      Z,INT20         ;NO,INT20
2085    CD 2096             CALL    SOF             ;SOUND OFF
2088    18 D2               JR      INTR            ;
                    ;       SOUND ON
208A    3E 07       SON:    LD      A,07H           ;LOAD CONTROL WORD
                                                    ;DISABLE INTERRUPT
                                                    ;TIMER MODE
                                                    ;PRESACALER=16
                                                    ;LOAD TIME CONSTANT
208C    D3 09               OUT     (CTC1),A        ;LOAD IT TO CTC 1
208E    3E 60               LD      A,96            ;LOAD TIME CONSTANT
2090    D3 09               OUT     (CTC1),A        ;
2092    36 03               LD      (HL),03H        ;SET SOUND ON TIME AND
                                                    ;WAIT FOR 500 MS
2094    18 06               JR      SOF0            ;
                    ;       SOUND OFF
2096    3E 03       SOF:    LD      A,03H           ;LOAD CONTROL WORD
2098    D3 09               OUT     (CTC1),A        ;RESET CTC 1
209A    36 02               LD      (HL),02H        ;SET SOUND OFF TIME AND
                                                    ;WAIT FOR 500 MS
209C    AF          SOF0:   XOR     A               ;CLEAR 500 MS COUNTER
209D    32 2044             LD      (COUNT),A       ;
20A0    C9                  RET                     ;
                    ;
                    ;       UP 500 MS COUNTER
20A1    3A 2044     UPC:    LD      A,(COUNT)       ;UP 500 MS COUNTER
20A4    3C                  INC     A               ;
20A5    32 2044             LD      (COUNT),A       ;
20A8    FE 32               CP      50              ;500 MS EXPIRED ?
20AA    C9                  RET                     ;
                    ;
                    ;
20AB                        DS      64              ;STACK AREA
20EB                STACK   EQU     $               ;
                    ;
                            END
```

3.3 Z80 DMA編

▶Z80 Direct Memory Access◀

DMAとは何か？　なぜDMAを必要とするか？

DMAというのは，CPUの動作を介さずにメモリとI/O間で，データ転送を行うために編み出された手法であり，その目的はデータ転送の高速化とプログラムの効率化にあります．ではDMAを用いるとなぜデータ転送の高速化がはかれ，プログラムの効率化がはかれるのでしょう．

これらのことがらを説明する前に，まずDMAではないデータ転送についてふりかえってみます．以下，本稿ではDMAではないデータ転送のことを，プログラム・データ転送と呼び，DMAによるデータ転送のことを，DMAデータ転送と呼ぶことにします．

プログラム・データ転送

図1にI/Oからデータを読み取り，これをメモリに書き込む場合のプログラム・データ転送の概念図を示します．この例ではプログラムはI/Oのステータス・レジスタの内容をインプット命令により読み込み，ステータスがデータ・レディであれば，再度インプット命令により，I/Oのデータ・レジスタからデータを読み取り，これをメモリに書き込みます．

以上のことがらを行うプログラム例を，Prog❶に示します．このプログラムは，Z80 CPUを使うことを前提にしていますが，これ以降に示すすべてのプログラムも，Z80 CPUを前提にしています．

Prog❶は，まずバッファ・ポインタ(HL)，バイト・カウンタ(B)，ならびにデータ・ポート・アドレス(C)をセットしたのちに，ステータス・レジスタの内容を読み，データ・レディ・ビット(ビット7)が立つのを待ちます．このビットが“1”になると，INI命令によりデータ・レジスタの内容を読み取り，データをメモリにストアします．

ここでステータス・レジスタのレディ・ビットはデータ・レジスタの内容を読むことにより，クリヤされ

〈Prog❶〉 プログラム・データ転送の例

```
                                ;
                                ;        PROGRAM DATA TRANSFER
                                ;
0000                            STATUS   EQU      00H          ;STATUS REGISTER
0001                            DATA     EQU      01H          ;DATA REGISTER
                                ;
0000      21 0011               BEGIN:   LD       HL,BUFFER    ;SET UP BUFFER POINTER
0003      06 00                          LD       B,0          ;SET UP BYTE COUNTER
0005      0E 01                          LD       C,DATA       ;LOAD DATA ADDRESS
0007      DB 00                 WAIT:    IN       A,(STATUS)   ;(11) READ STATUS
0009      17                             RLA                   ;(04) DATA READY ?
000A      28 FB                          JR       Z,WAIT       ;(07) NO,WAIT
000C      ED A2                          INI                   ;(16) READ DATA AND
                                                               ;     WRITE IT INTO
                                                               ;     MEMORY.
000E      C2 0007                        JP       NZ,WAIT      ;(10) READ 256 ?
                                                               ;     NO,WAIT
                                ;
                                ;
                                ;
0011                            BUFFER:  DS       256          ;DATA BUFFER
                                ;
                                         END
```

るものと仮定します．その後，プログラムは次のデータを求めて，再度ステータス・レジスタをチェックしにいきます．以上の動作が256バイトのデータをI/Oから受け取るまで続きます．

この方法により，1バイトのデータをI/Oから読み取り，それをメモリに書き込むのに要する時間の最小値は，Prog❶の0007_{16}番地から$000E_{16}$番地までの命令実行時間の和となり，48ステートとなります．このステート数は，Z80 CPUを4 MHzで動作させた場合，12μsとなります．したがって転送速度の最大値は，約83Kバイト／秒です．

図1の概念に対して少し変更を加え，プログラムによりデータ・レディをチェックするのをやめ，データ・レディでないときには，I/OがCPUに対してウェイトをかけるようにすると，転送速度の向上がはかれます．図2に，ウェイトを用いたプログラム・データ転送の概念図を示します．

この例では，I/Oはインプット命令がくると，データの用意ができるまで，$\overline{\text{WAIT}}$信号を“L”にしてCPUの動作をストップさせます．I/Oはデータの用意ができると，$\overline{\text{WAIT}}$信号を“H”にし，CPUを走らせます．

これによりプログラムはデータ・レジスタからデータを読み取り，それをメモリに書き込みます．I/Oはデータ・レジスタからデータを読み取られ，次にインプット命令がきたときに，データの用意ができていない場合には，再度$\overline{\text{WAIT}}$信号を“L”にします．

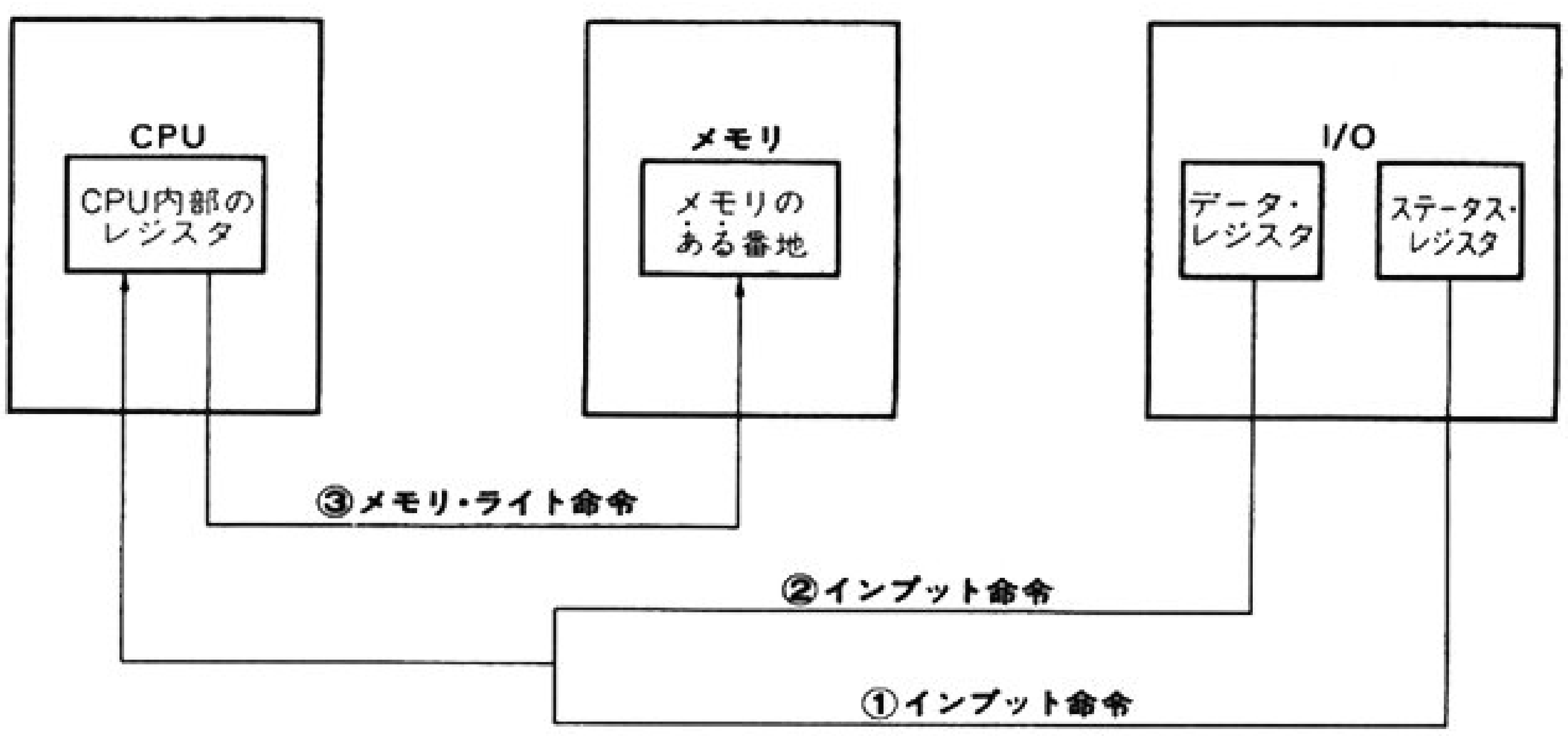
以上のことがらを行うプログラムを，Prog❷に示します．

このプログラムでは，データ転送を行うための命令はINIRのみです．INIR命令の実行時間は21ステートですので，Z80 CPUを4 MHzで動作させれば，この方式によるデータ転送時間の最小値は，1バイト当たり5.25μsとなり，データ転送速度の最大値は約190Kバイト／秒となります．

したがって，この方式によれば図1のやりかたに比べて2倍以上速い転送速度が達成できます．なお，細かい話になりますが，INIR命令の実行時間は，レジスタBの値がゼロになったときには16ステートです．しかしこの例では転送速度におよぼす影響はゼロに等しいので，一律に21ステートとして計算しました．

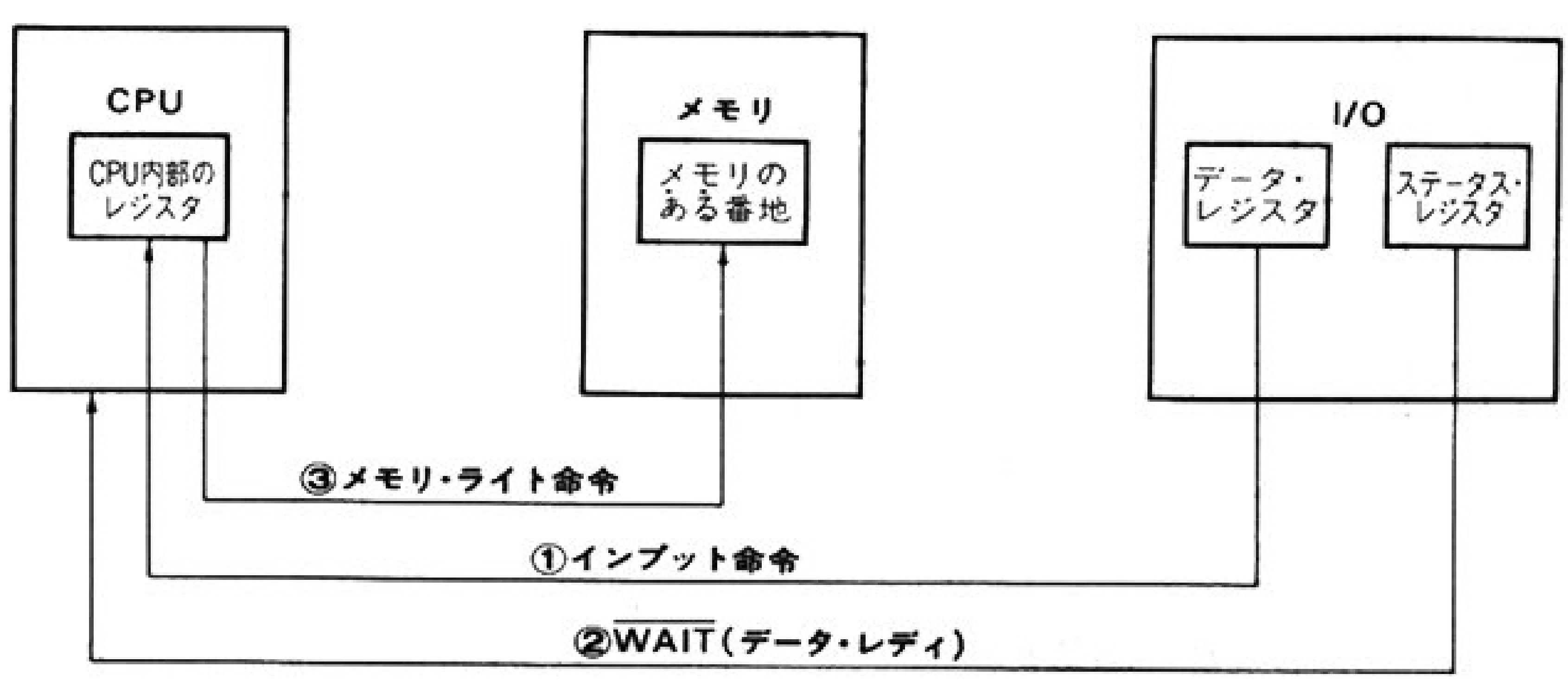
＊

以上の話はプログラム・データ転送を，データ転送速度の面からのみながめていましたが，次にプログラム効率の面からながめてみます．Prog❶およびProg❷で示したプログラム・データ転送では，I/Oとデー



〈図1〉
プログラム・データ転送の概念図

① I/Oのステータス・レジスタの内容を読み，データ・レディであるか否かをチェックする．
② データ・レディであれば，I/Oからデータを読み込む．
③ 読み込んだデータをメモリに書き込む．



〈図2〉
ウェイト($\overline{\text{WAIT}}$)を用いた，プログラム・データ転送の概念図

① インプット命令がくると，I/Oはデータ・レディになるまで$\overline{\text{WAIT}}$を“L”にする．
② データ・レディになると，I/Oは$\overline{\text{WAIT}}$を“H”にする．
③ I/Oからデータを読み込み，それをメモリに書き込む．

〈Prog ❷〉 ウェイトを用いたプログラム・データ転送の例

```
                    ;
                    ;       PROGRAM TRANSFER WITH WAIT
                    ;
0000                CTL     EQU     00H             ;INITIATE TRANSFER
0001                DATA    EQU     01H             ;DATA REGISTER
                    ;
0000   21 000B      BEGIN:  LD      HL,BUFFER       ;SET UP BUFFER POINTER
0003   06 00                LD      B,0             ;SET UP BYTE COUNTER
0005   0E 01                LD      C,DATA          ;SET DATA ADDRESS
0007   D3 00                OUT     (CTL),A         ;INITIATE TRANSFER
0009   ED B2                INIR                    ;DO TRANSFER
                    ;
                    ;
000B                BUFFER: DS      256             ;DATA BUFFER
                    ;
                            END
```

タの授受を行っている期間，CPU はそれだけに専念させられます．

つまり，**Prog ❶** ではI/Oのステータス・レジスタの内容をチェックし，データ・レジスタからデータを読むことだけに CPU が使われています．したがって，I/O がデータの用意をできるのがたとえば 1 ms に 1 回とすると，約256 ms の期間がデータ転送に費やされることになります．

このことは **Prog ❷** に対してもいえます．つまり，**Prog ❶** および **Prog ❷** で示したようなデータ転送方式では，同時並行処理ができないので，CPU はデータ転送専用マシン，もっと大げさにいえばステータス・チェック専用マシンになってしまいます．

このことを改善するために考え出されたのが割り込みです．図 3 に割り込みを用いたプログラム・データ転送の概念図を示します．I/O はデータの用意ができると，CPU に対して割り込みをかけ，割り込みルーチンはI/Oからデータを読み込み，それをバッファにストアします．I/O はデータを読み取られると割り込みを解除します．

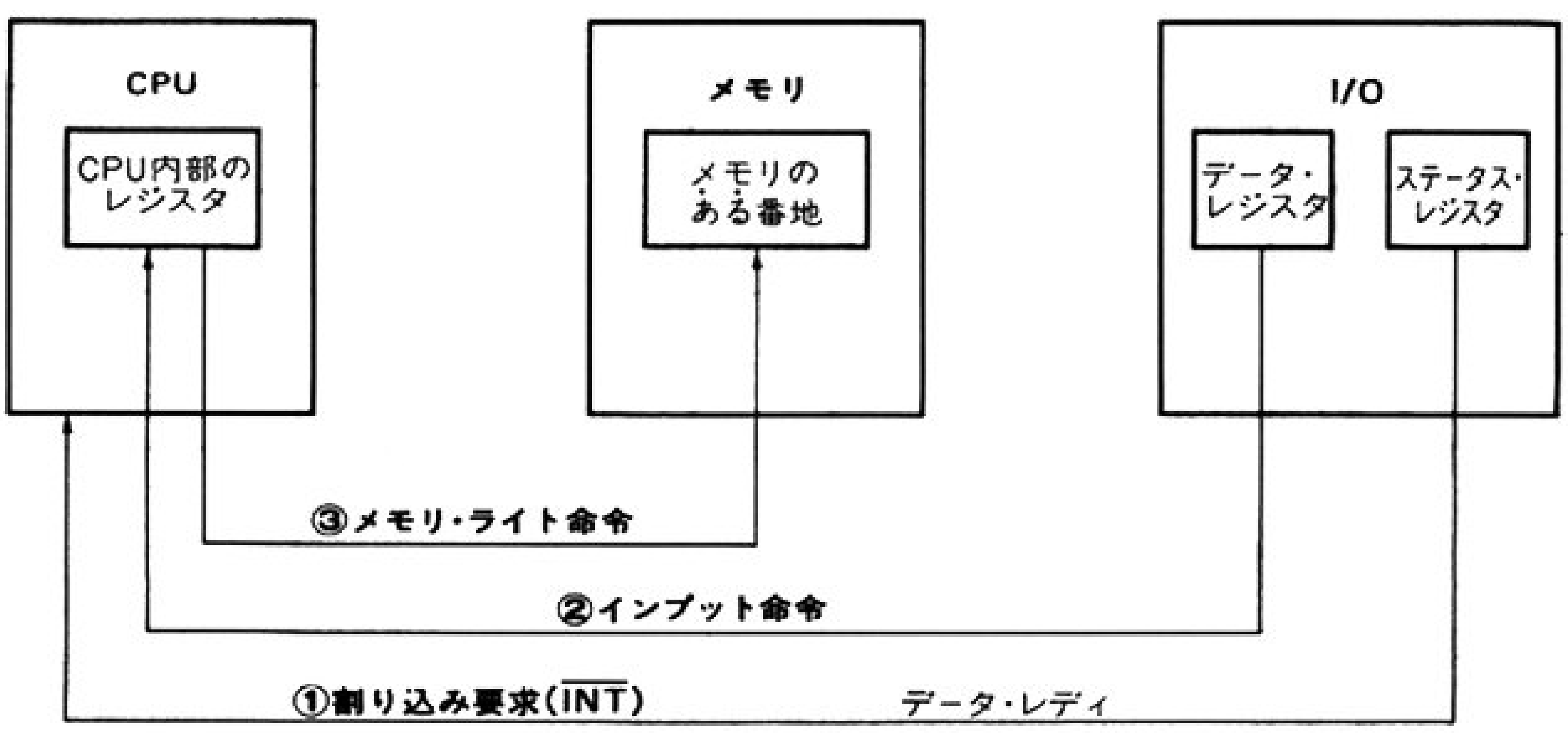
この例ではI/Oのステータス・レジスタのデータ・レディをプログラムがチェックするのではなく，割り込みによりI/Oがプログラムに知らせるやりかたをとっています．したがって，プログラムは割り込みによりデータ・レディであることを知らされるまで，他のことをやっていてよいことになります．データ転送と同時に"他のことをやる"ことを同時並行処理と呼びます．

図 4 に割り込みによるプログラム・データ転送と同時並行処理の流れを示します．この処理では 256 バイトのバッファを二つ（バッファ 1 およびバッファ 2）持ち，一方のバッファにデータを詰め込んでいる間に，他方のバッファのデータを処理します．これを交換バッファリング方式と呼びます．

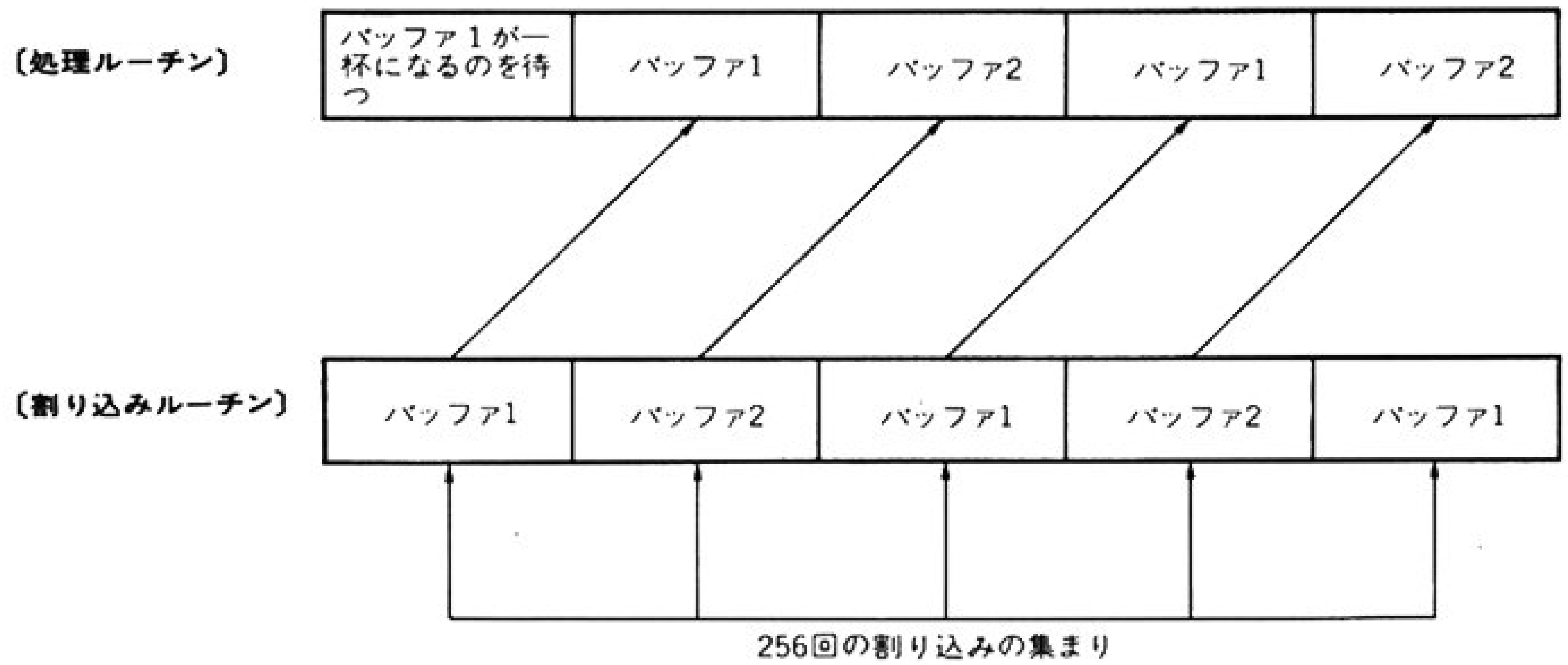
一番始めには両バッファとも空なので，プログラムはバッファ 1 が一杯になるのを待ちます．このバッファ 1 には割り込みルーチンがプログラム・データ転送により，データを詰めこみます．そしてバッファ 1 が一杯になると，プログラムはバッファ 1 のデータの処理を行い，これと並行して割り込みルーチンはバッファ 2 にデータを詰めこみます．

そしてバッファ 1 のデータの処理が終わると，バッ

〈図 3〉
割り込みによるプログラム・データ転送の例



① I/O はデータ・レディになると CPU に対して割り込み要求を出す．
② 割り込みルーチンは，I/O からデータを読み込む．
③ 読み込んだデータを，メモリに書き込む．



<図4>
割り込みによるプログラム転送と交換バッファリングによる同時並行処理

ファ2が一杯になるのを待ち，バッファ2が一杯になると，その処理を行うと同時に，割り込みルーチンはバッファ1にデータを詰めこみ始めます．以上の過程を行うプログラムを，**Prog❸**に示します．

割り込みを用いることにより，同時並行処理が可能となり，プログラムの効率を上げることができることがわかりましたが，割り込みを用いた場合，1バイトのデータをI/Oから読み取り，それをメモリに書き込むに要する時間は，単純なプログラム・データ転送(**Prog❶**および**Prog❷**)に比べて，大幅に大きくなります．

Prog❸では，交換バッファリングを行っているため，プログラムが複雑になっていますので，割り込みが発生してから，I/Oのデータを読み，それをメモリにストアし，バッファのコントロールをして，割り込みルーチンを抜けるのに(**Prog❸**のINT：からRETI命令まで)，実に239ステートを要します．

Prog❸ほど複雑にしなかったとしても，割り込みを用いる場合は，必ずレジスタの退避／回復，割り込みの再イネーブル，ならびにRETIを必要とします．**Prog❹**にもっとも単純化した，割り込みルーチンの例を示します．この場合には1バイト当たりに要する時間は50ステートとなります．この値は**Prog❶**の場合の最小時間とほぼ同じです．

■プログラム・データ転送のまとめ

以上，「DMAデータ転送とはどんなものであるか」ということを説明するための予備知識として，プログラム・データ転送について比較的詳しく述べてきましたが，ここでプログラム・データ転送についてまとめておきます．

❶ **Prog❶**のようなポールド・モード(Polled Mode)のプログラム・データ転送では，1バイトのデータを入力するのに要する時間の最小値は48ステートであり，Z80 CPUを4MHzで動作させた場合，12μsとなる．転送速度の最大値は83Kバイト／秒である．

❷ **Prog❷**のように$\overline{\text{WAIT}}$を用いたプログラム・データ転送を行うことにより，転送速度を190Kバイト／秒くらいまで上げることができ，1バイトのデータを入力するのに要する時間は,18ステートである．

❸ **Prog❶**および**Prog❷**のような方式では，CPUはデータ転送マシンになるので，CPUの使用効率が低下する．

❹ 割り込み駆動型でかつ交換バッファリングを行う

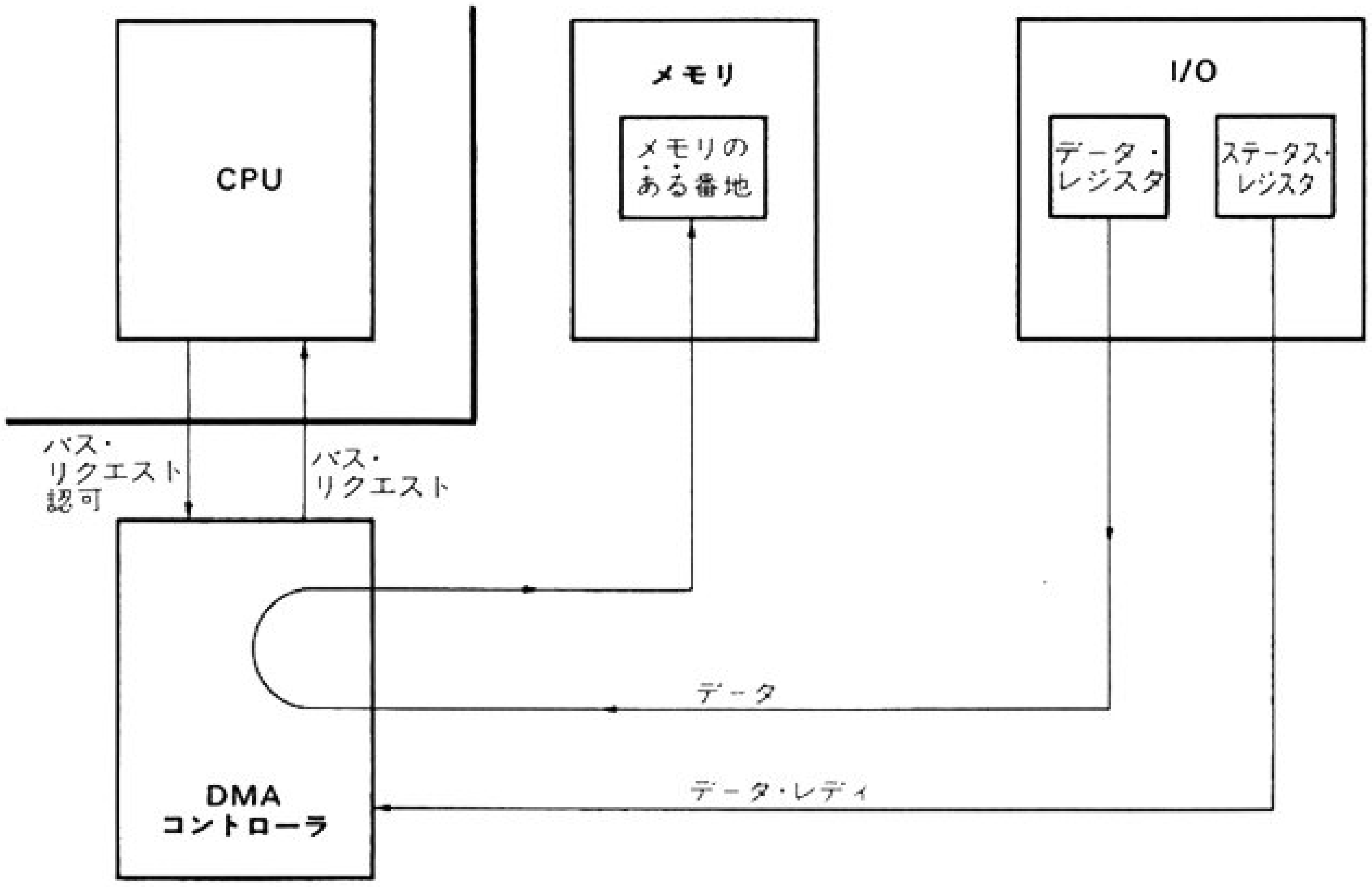
```
;
;       SIMPLIFIED INTERRUPT ROUTINE
;
INT:    EXX                         ;SAVE REGISTERS
        EX      AF,AF'              ;SAVE PSW
        INI                         ;READ DATA AND STORE IT
        EX      AF,AF'              ;RESTORE PSW
        EXX                         ;RESTORE REGISTERS
        EI                          ;RE-ENABLE INTERRUPT
        RETI                        ;
;
        END
```

<Prog❹>
単純化した割り込みルーチン

```
                        ;
                        ;       PROGRAM TRANSFER WITH INTERRUPT
                        ;
0000                    DATA    EQU     00H             ;I/O DATA REG. ADDRESS
                        ;
                        ;       BUFFER CONTROL TABLE EQUATES
0000                    BCT$BA  EQU     0               ;BUFFER ADDRESS
0002                    BCT$BP  EQU     BCT$BA+2        ;BUFFER POINTER
0004                    BCT$BC  EQU     BCT$BP+2        ;BYTE COUNTER
0006                    BCT$LK  EQU     BCT$BC+2        ;BUFFER LINK
                        ;
0000    31 02C8         BEGIN:  LD      SP,STACK        ;SET STACK POINTER
0003    3E 00                   LD      A,0             ;LOAD IR
0005    ED 47                   LD      I,A             ;
0007    ED 5E                   IM      2               ;SET MODE 2
0009    21 0086                 LD      HL,BUFF1        ;LOAD BUFFER 1 ADRS
000C    11 0187                 LD      DE,BUFF2        ;LOAD BUFFER 2 ADRS
000F    FB                      EI                      ;ENABLE INTERRUPT
                        ;       CHECK IF BUFFER READY
0010    7E              LOOP:   LD      A,(HL)          ;LOAD BUFFER FLAG
0011    B7                      OR      A               ;BUFFER READY ?
0012    28 FC                   JR      Z,LOOP          ;NO,LOOP
0014    D5                      PUSH    DE              ;SAVE BUFFER ADDRESSES
0015    E5                      PUSH    HL              ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;       PROCESS DATA IN THE BUFFER POINTED BY HL
                        ;
                        ;
                        ;
0016    E1                      POP     HL              ;RESTORE BUFFER ADRSES
0017    D1                      POP     DE              ;
0018    36 00                   LD      (HL),0          ;SET BUFFER FREE
001A    EB                      EX      DE,HL           ;EXCHANGE BUFFER
001B    C3 0010                 JP      LOOP            ;GO AND PROCESS NEXT
                                                        ;BUFFER
                        ;
                        ;       INTERRUPT ROUTINE
001E    D9              INT:    EXX                     ;EXCHANGE REGISTERS
001F    08                      EX      AF,AF'          ;EXCHANGE PSW
0020    DD E5                   PUSH    IX              ;SAVE IX
0022    DD 2A 0084              LD      IX,(BCTBLP)     ;LOAD BCTBL ADDRESS
0026    DD 6E 02                LD      L,(IX+BCT$BP)   ;LOAD BUFFER POINTER
0029    DD 66 03                LD      H,(IX+BCT$BP+1) ;
002C    DD 46 04                LD      B,(IX+BCT$BC)   ;LOAD BYTE COUNTER
002F    0E 00                   LD      C,DATA          ;LOAD I/O ADDRESS
0031    ED A2                   INI                     ;READ DATA FROM I/O
                                                        ;AND STORE IT INTO BUF.
0033    28 0B                   JR      Z,INTO          ;JUMP IF BUFFER FULL
                        ;       UPDATE BUFFER CONTROL TABLE
0035    DD 75 02                LD      (IX+BCT$BP),L   ;UPDATE BUFFER POINTER
0038    DD 74 03                LD      (IX+BCT$BP+1),H ;
003B    DD 70 04                LD      (IX+BCT$BC),B   ;UPDATE BYTE COUNTER
003E    18 2A                   JR      INT1            ;
```

```
                                ;
                                ;       SET BUFFER READY
0040    DD 6E 00        INT0:   LD      L,(IX+BCT$BA)      ;LOAD BUFFER FLAG ADRS
0043    DD 66 01                LD      H,(IX+BCT$BA+1)    ;
0046    36 FF                   LD      (HL),0FFH          ;SET BUFFER READY
                                ;       EXCHANGE BUFFER
0048    DD 6E 06                LD      L,(IX+BCT$LK)      ;LOAD NEXT BCTBL ADRS
004B    DD 66 07                LD      H,(IX+BCT$LK+1)    ;
004E    E5                      PUSH    HL                 ;HL --> IX
004F    DD E1                   POP     IX                 ;
0051    DD 6E 00                LD      L,(IX+BCT$BA)      ;LOAD NEW BUF ADRS
0054    DD 66 01                LD      H,(IX+BCT$BA+1)    ;
0057    7E                      LD      A,(HL)             ;BUFFER FREE ?
0058    B7                      OR      A                  ;
0059    20 16                   JR      NZ,OVRN            ;NO,OVER RUN ERROR
                                ;       INITIALIZE BUFFER CONTROL TABLE
005B    23                      INC     HL                 ;SKIP FLAG
005C    DD 75 02                LD      (IX+BCT$BP),L      ;SET BUFFER POINTER
005F    DD 74 03                LD      (IX+BCT$BP+1),H    ;
0062    DD 36 04 00             LD      (IX+BCT$BC),0      ;RESET BYTE COUNTER
0066    DD 22 0084              LD      (BCTBLP),IX        ;SET NEW BCTBL ADRS
                                ;       RETURN FROM INTERRUPT
006A    DD E1           INT1:   POP     IX                 ;RESTORE IX
006C    08                      EX      AF,AF'             ;RESTORE PSW
006D    D9                      EXX                        ;RESTORE REGISTERS
006E    FB                      EI                         ;RE-ENABLE INTERRUPT
006F    ED 4D                   RETI                       ;
                                ;
                                ;       DATA OVER RUN ERROR
0071    76              OVRN:   HALT                       ;ERROR STOP
                                ;
0072    001E            INTV:   DW      INT                ;INTERRUPT VECTOR
                                ;
                                ;       BUFFER CONTROL TABLE
0074    0086            BCTBL1: DW      BUFF1              ;BUFFER 1 ADDRESS
0076    0087                    DW      BUFF1+1            ;BUFFER POINTER
0078    0000                    DW      0                  ;BYTE COUNTER
007A    007C                    DW      BCTBL2             ;BUFFER LINK
                                ;
007C    0187            BCTBL2: DW      BUFF2              ;BUFFER 2 ADDRESS
007E    0188                    DW      BUFF2+1            ;BUFFER POINTER
0080    0000                    DW      0                  ;BYTE COUNTER
0082    0074                    DW      BCTBL1             ;BUFFER LINK
                                ;
0084    0074            BCTBLP: DW      BCTBL1             ;BCTBL POINTER
                                ;
                                ;       BUFFERS
0086    00              BUFF1:  DB      0                  ;BUFFER 1 FLAG
0087                            DS      256                ;DATA BUFFER
0187    00              BUFF2:  DB      0                  ;BUFFER 2 FLAG
0188                            DS      256                ;DATA BUFFER
                                ;
0288                            DS      64                 ;STACK AREA
02C8                    STACK   EQU     $                  ;
                                ;
                                ;
                                        END
```

〈図5〉
DMA データ転送の概念図



I/O がデータ・レディであることを示すと，DMA コントローラは CPU を隔離し，I/O データ・レジスタの内容を読み取って，それをメモリに書き込む

ことにより，CPU の使用効率を上げることができるが，１バイトのデータを入力するのに大きな時間を要するため，転送速度は低下する．

❺ 割り込み駆動型の場合には，割り込みルーチンにおいて，レジスタの退避／回復を要するため，ポールド・モードの場合に比べて，転送速度は低くなる．

DMA データ転送とは何か？

DMAデータ転送はプログラム・データ転送の持つ二つの欠点，すなわち，

❶ CPU の使用効率が悪い

❷ データ転送速度がさほど速くない

を克服するために考案された方法であり，割り込みとともにコンピュータ史上，最高峰に位置する発明です．

DMA というデータ転送方式が産み出された背景には次のようなことがらがあります．すなわち，

❶ I/O とメモリ間のデータ転送などという単純な作業に高価な CPU（昔は CPU は非常に高価であった）を介在させる必要はない．CPU はもっと有意義な演算に専念すべきである．

❷ ドラム，磁気テープなどの高速な I/O が動き出すと，プログラム・データ転送を使用していると，CPU はデータ転送だけに集中せねばならず，肝心な演算を行う時間がほとんど残らない．また I/O がさらに高速の場合には，CPU がやっていたのではデータ転送が間に合わなくなる（オーバラン）ようなことがありうる．

同じようなことが，現在のマイクロプロセッサの世界においてもいえます．倍密度フロッピー・ディスクのデータ転送速度は 500K ビット／秒であり，これをバイトに直すと 62.5K バイト／秒です．この転送速度は前項のプログラム転送でも対処できる範囲にありますが，CPU の動作時間の大半をデータ転送にとられてしまいます．

では今はやりのウィンチェスタ・タイプの磁気ディスク装置を考えてみると，これらのディスクのデータ転送速度は１メガ・バイト／秒くらいあります．これは１バイト当たり，1 μs で転送されることを意味し，プログラム・データ転送では，どうあがいてみても不可能です．

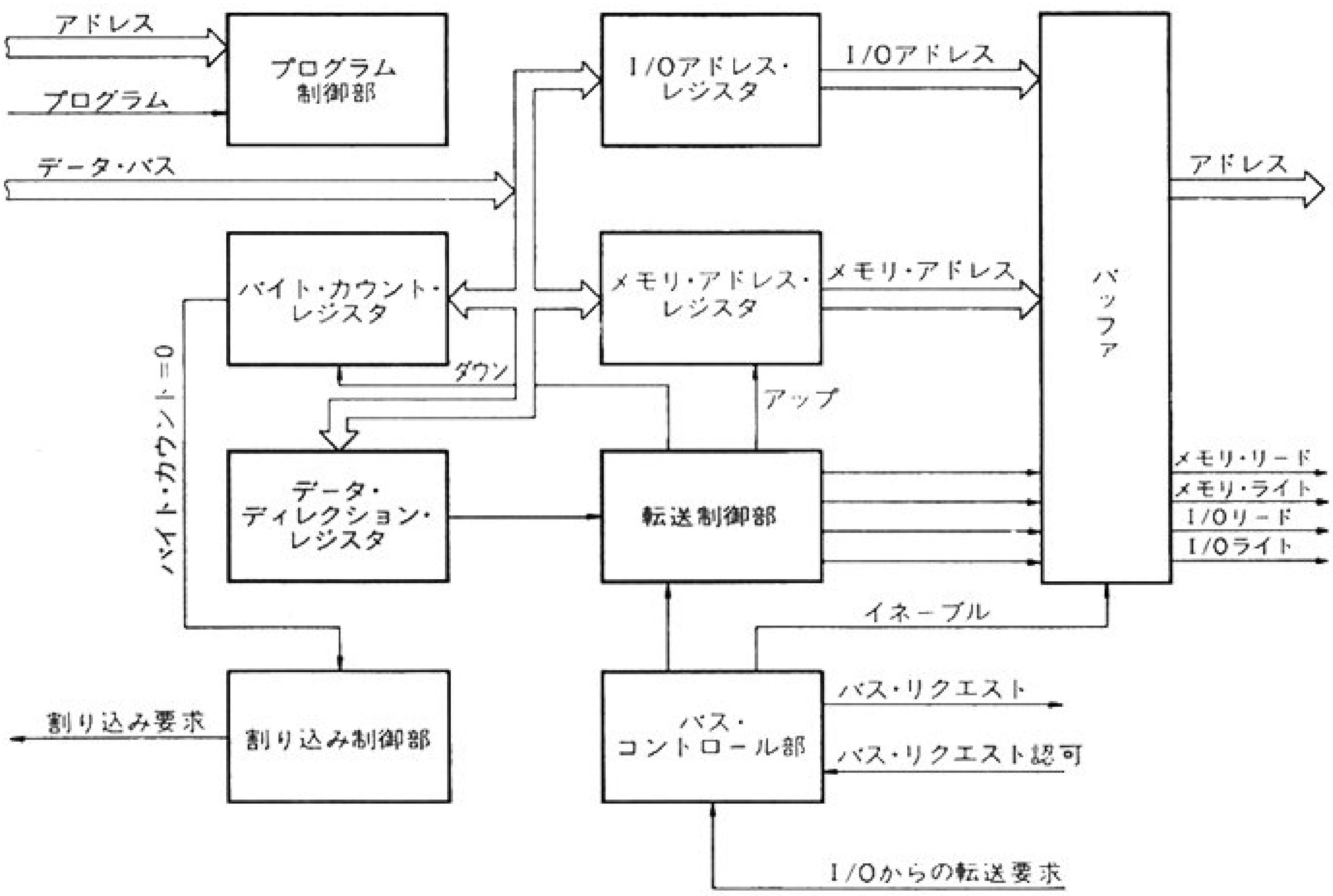
以上より，プログラム・データ転送を行っていたのでは処理効率が低下してしまったり，I/O が速すぎて間に合わないような場合には，何らかの手段を講じる必要があります．この手段がこれから説明する，DMA データ転送です．

DMA データ転送

DMA ということばは，Direct Memory Access の各頭文字をとった略語であり，I/O とメモリ間のデータ転送を CPU を介さずに，直接的にやってしまうという意味です．

CPU のやっかいにならずに I/O とメモリ間でデータ転送をするのですが，データ転送を I/O とメモリだけに委せておいては，のけものにされた CPU が文句をいうのは当然です．つまり CPU はデータ転送に必要なバスの管理権は自分にあると思っているからです．そこで I/O，メモリと CPU の間の仲をとりもつ仲人役が必要となり，これを通常 DMA コントローラと呼びます．

図5に DMA データ転送の概念図を示します．I/O はデータの用意ができると，DMA コントローラに対してデータ・レディ信号を出します．これを受けると，D



〈図6〉
一般的なDMAコントローラの構成図

MAコントローラは，CPUに対して「ちょっとバスを貸してください」という意味の，バス・リクエストを出します．

このリクエストがあるとCPUはその時点で行っている動作の区切りのいいところでストップし，バスの状態をハイ・インピーダンスにすることにより，自分自身をバスから論理的に切り離した後に，バス・リクエスト認可（バス・アクノウリッジ：Bus Acknowledge）をDMAコントローラに対して出し，バスの使用権をDMAコントローラに与えます．

バスの使用権を与えられたDMAコントローラは，I/Oからデータを読み取りそれをメモリに書き込みます．データを読み取られたI/Oはデータ・レディを落とします．そしてデータ転送を終えると，DMAコントローラはバス・リクエストを引っこめます．バス・リクエストがなくなると，CPUはバス・リクエスト認可を落とし，中断していた仕事を続行します．

メモリからI/Oへデータを転送する場合も同じです．データが必要になると，I/Oはデータ・レディをDMAコントローラに出します．DMAコントローラは，前の場合と同じようにしてバスの制御権をCPUからもらい，メモリからデータを読み取り，それをI/Oに渡します．

以上がDMAデータ転送の概念ですが，“なぜDMAデータ転送がプログラム・データ転送に比べて速いか”といいますと，DMAコントローラがデータ転送専用に設計されたハードウェアだからです（専用のハードウェアを設ければ，ものごとを速く行うことができるのは当たり前であり，それほどおどろく必要はないのかもしれません）．本稿で説明するZ80 DMAの場合，メモリとI/O間のデータ転送を，1バイト当たり2クロック期間でやってしまいます．したがって4 MHzのクロックを用いると，0.5μsで1バイトのデータ転送ができることになります（2Mバイト／秒）．

*

ではDMAコントローラのハードウェアはどのような構成をしていればよいのか考えてみましょう．まずDMAコントローラとして最低限必要な機能をあげると，以下の8点をあげることができます．すなわち，

① I/Oからのデータ転送要求を認識する機能
② CPUに対してバス・リクエストをし，CPUの認可を識別する機能
③ メモリおよびI/Oをアドレッシングする機能
④ メモリおよびI/Oに対して読み書きの指示を与える機能
⑤ 転送すべきデータの量（バイト・カウント）を管理する機能
⑥ 転送するデータの方向（メモリ → I/O, I/O → メモリ）を制御する機能
⑦ データ転送の終了をプログラムに知らせる機能
⑧ DMAコントローラをプログラミングする機能

これらの機能を持たせると，一般的なDMAコントローラを構成することができます（図6）．各構成部分の働きは次のとおりです．

・プログラム制御部

DMAデータ転送に先立って，データ転送に関連するメモリのアドレス，I/Oアドレス，バイト・カウント，ならびにデータの方向などをプログラミングする必要

があります．プログラム制御部はデータ・バス上のデータを必要な場所にセットする制御を行います．

・バス・コントロール部

I/O からデータ転送要求があると，CPU に対してバス・リクエストを出し，これが認められるとバッファをイネーブルします．

・I/O アドレス・レジスタ

データ転送の対象となる I/O のアドレスを保持するレジスタです．

・メモリ・アドレス・レジスタ

データ転送の対象となるメモリのアドレスを保持するレジスタであり，1バイトのデータ転送が終わるごとに1だけ進められる，アップ・カウンタです．

・バイト・カウント・レジスタ

転送すべきデータの数を保持するレジスタで，1バイト・データの転送が終わるごとに，1ずつ減じられるダウン・カウンタです．

・データ・ディレクション・レジスタ

データ転送の方向（メモリ → I/O, I/O → メモリ）を指定するためのレジスタです．

・転送制御部

データ・ディレクション・レジスタの状態にしたがって，メモリ・リード／ライト信号，I/O リード／ライト信号を出力し，1バイトの転送が終わるごとに，メモリ・アドレス・レジスタとバイト・カウント・レジスタの内容を更新します．

・割り込み制御部

データ転送が完了したこと（バイト・カウント＝0）を，プログラムに知らせるために，CPU に対して割り込み要求を出します．

・バッファ

CPU が動作しているときには，DMA コントローラはバスに対して何の権利もありませんので，このときにはアドレス・ライン，およびメモリや I/O のコントロール・ラインをハイ・インピーダンスにする必要があります．また DMA データ転送を行うときには，このバッファをイネーブルし，CPU に代わってバスの制御を行います．

*

以上で"DMA とは何か？"という項を終わりますが，ここで"DMA はどのような場合に，なぜ有効であるのか"をまとめておきます．

❶ ディスクなどの高速な I/O を使用するときには，プログラム・データ転送をしていたのでは，転送速度が追いつけず，データのオーバラン状態を発生する．このようなときには DMA により高速データ転送を行わざるを得ない．

❷ フロッピー・ディスクやプリンタなどとデータ転送をするときには，プログラム・データ転送でもやっていけるが，CPU はかなりの時間をデータ転送のためにさかなければならない．このような場合にも DMA データ転送を行うことにより，CPU の負荷を相当軽減できる．

とくにデータ転送終了時の DMA 割り込みを用いることにより，I/O オペレーションと演算の同時並行処理が可能となり，CPU の効率を上げることができる．

Z80 DMA

Z80 DMA は Z80 ファミリの DMA コントローラであり，動作速度に応じて Z80 DMA(2.5MHz) と Z80A DMA(4 MHz) がありますが，以下の説明では Z80 DMA で通すことにし，動作速度の違いなどで両者を分けて説明をする必要のある場合のみ，Z80A DMA の名前を用いることにします．したがって，とくにことわりのない限り Z80 DMA といった場合，Z80A DMA をも含むこととします．

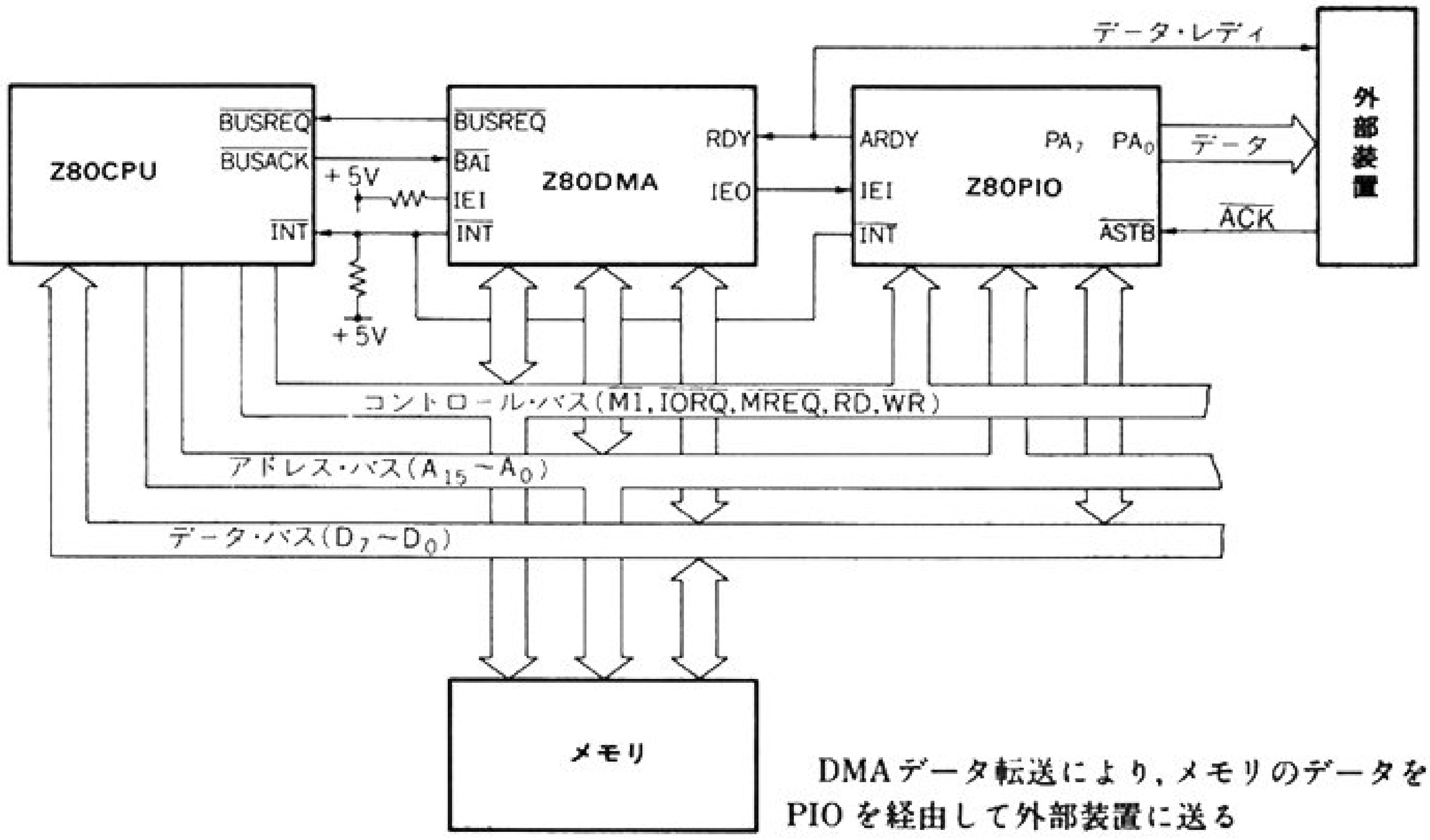
さて，Z80 DMA は次のような特徴を持っています．

・3種類のデータ転送形式
  ① データ転送（トランスファ）
  ② サーチ
  ③ データ転送，およびサーチ（トランスファ／サーチ）
・3種類のデータ転送モード
  ① バイト・モード
  ② バースト・モード
  ③ コンティニュアス・モード
・データの転送方向を自由に設定できる
・オート・リスタート機能
・デイジー・チェイン式のベクトル割り込み発生機能
・レディ信号の有効極性をプログラミングできる
・デイジー・チェイン式のバス優先順位制御機能
・バス・タイミングが可変
・外部アドレス・ラッチを必要としない

Z80 DMA の使用例

Z80 DMA の機能，および動作ならびにプログラミングなどについて説明する前に，Z80 DMA が Z80 マイクロコンピュータ・システム中でどのような使われかたをするのかを述べ，Z80 DMA の概要をつかんでいただこうと思います．

図7に Z80 DMA を用いた簡単なシステムを示します．このシステムは DMA データ転送により，メモリのデータを Z80 PIO に送り，ついでそのデータを外



DMAデータ転送により，メモリのデータをPIOを経由して外部装置に送る

〈図7〉 Z80 DMAを用いた簡単なシステム

部装置（たとえばプリンタ）に送ります．

この例ではZ80 DMAはメモリからI/O(Z80 PIO)へデータ転送をするようにプログラミングされているものとし，Z80 PIOはモード0にプログラミングされているものとします．またZ80 DMAのレディ(RDY)は“L”で有効となるようにプログラミングされているものとします．

以上の仮定のもとに，このシステムの動作を説明します．Z80 DMAがイネーブルされると，Z80 PIOのARDYは“L”ですので，Z80 DMAのレディが有効となり，DMA動作が開始され，CPUに対して$\overline{\mathrm{BUSREQ}}$が出されます．この$\overline{\mathrm{BUSREQ}}$を検出すると，CPUは自分の都合の良い時点に動作を停止し，$\overline{\mathrm{BUSACK}}$を“L”にします．

$\overline{\mathrm{BUSACK}}$が“L”になると，Z80 DMAはアドレス・バス，データ・バス，ならびにコントロール・バスを使用して，メモリからデータを読み取り，それをZ80 PIOに送ります．このデータを受け取ると，Z80 PIOはARDYを“H”にし，同じデータをPA_7～PA_0上にセットします．

このARDYが“H”になると，Z80 DMAのRDYが無効極性になりますので，Z80 DMAは$\overline{\mathrm{BUSREQ}}$を“H”にします．$\overline{\mathrm{BUSREQ}}$が“H”になると，CPUは$\overline{\mathrm{BUSACK}}$を“H”にし，中断していた仕事を続行します．

またARDYは外部装置に対しては，“PA_7～PA_0上にデータの用意ができている”ことを示しますので，外部装置はPA_7～PA_0上のデータを取り込み，ついで“データを受け取った”ことをZ80 PIOに知らせるために，$\overline{\mathrm{ACK}}$パルスを出力します．この$\overline{\mathrm{ACK}}$を受け取ると，Z80 PIOはARDYを“L”にし，再度Z80 DMAに対してデータ転送を要求します．

この過程はZ80 DMAにプログラミングされた，バイト・カウントがゼロになるまで繰り返され，バイト・カウントがゼロになると，Z80 DMAはCPUに対して割り込み($\overline{\mathrm{INT}}$)を発生し，データ転送の終了をプログラムに知らせます．

Z80 DMAの中身の説明もしないで，突然Z80 DMAの使いかたの話が出てきて，面くらわれた人もあると思いますが，前項の「DMAとは何か？」を読んで理解している読者ならば，以上の話のほとんどをおわかりいただけると思います．

もちろん，Z80 PIOに関する知識があることは前提条件です．Z80 DMAの機能は相当複雑ですので，個個の機能の話からはじめると，Z80 DMAの全貌を見失ってしまうおそれがあるので，まず一番ふつうの使用法を示したわけです．

Z80 DMAの構成

図8にZ80 DMAの構成図を示します．以下に各部の動作を説明します．

・バス制御部

プログラミング時には，Z80 DMAは単なるI/Oポートと同じになりますので，コントロール・バスはシステム・バス ➡ Z80 DMAの向きになります．またZ80 DMAのコントロール・レジスタに対して，プログラミング・データを書き込むときには，データ・バスはシステム・バス ➡ Z80 DMAの向きになり，コントロール・レジスタからデータ（ステータスなど）を読み取るときには，データ・バスの方向はその逆になります．

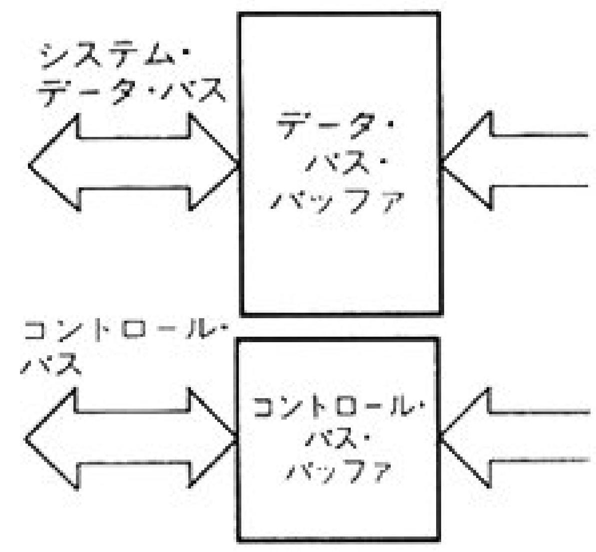
もちろん，このときにはZ80 DMAにはアドレス・バスの管理権はありませんので，アドレス・バッファはディスエーブルされます．

DMAデータ転送時には，コントロール・バス，およびアドレス・バスの管理はZ80 DMAに委(まか)されますので，コントロール・バス・バッファ，およびアドレス・バス・バッファはイネーブルされ，これらのバスの向きは，Z80 DMA ➡ システム・バスとなります．

データ・バスに関しては，メモリまたはI/Oからデータを読み取るタイミングでは，システム・バス ➡ Z80 DMAの向きとなり，データを一時内部にラッチします．メモリまたはI/Oにデータを書き込むタイミングでは，データ・バスの方向をZ80 DMA ➡ システム・バスの方向にし，ラッチしていたデータをシステム・データ・バス上に乗せます．

バス制御部は以上のようなバス方向制御を行うとともに，DMAデータ転送時にはデータの転送方向(メモリ ⟷ I/O)にしたがって，コントロール・バスを制御します．

〈図8〉
Z80 DMAの構成



〈図9〉 Z80 DMAのコントロール・レジスタ（ライト・レジスタ）の構成．ライト・レジスタ0～6に対して，WR0～WR6の名前が付けられている



またプログラミング時には，内部データ・バス上のデータを必要なコントロール・レジスタに書き込む管理をし，コントロール・レジスタの内容を読む場合には，必要なレジスタの内容を内部データ・バス上に乗せます．

このほか，バス制御部はDMA転送時において必要なことがら（アドレス・カウンタやバイト・カウンタの更新など）を管理します．

・優先順位制御部

この部分は割り込み優先順位とバス制御権の優先順位を管理します．ご存知のようにZ80システムでは，各LSI間の割り込み優先順位は，デイジー・チェイン方式により管理されています．

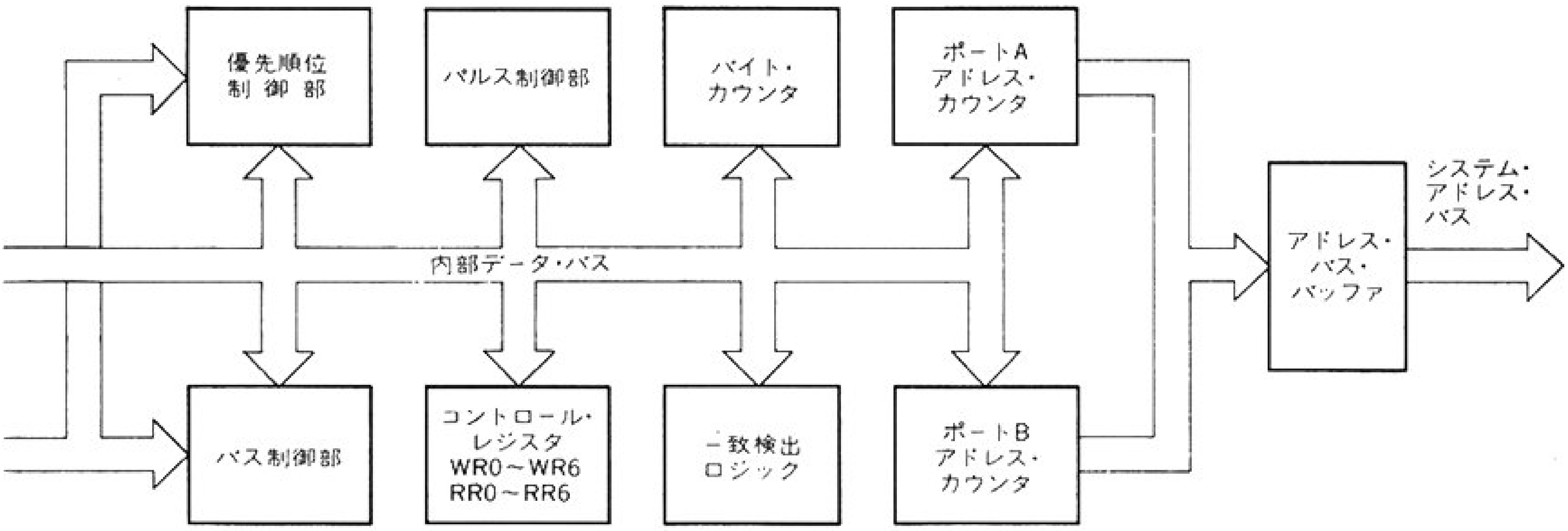
これと同じように，Z80 DMAでは，複数のDMAコントローラ間のバス制御権の配分をデイジー・チェイン方式により管理します．優先順位制御部はこれら二つのデイジー・チェインを管理する役目を果たします．

・コントロール・レジスタ

Z80 DMAのコントロール・レジスタは，それぞれ7個のライト・レジスタ(WR0～WR6)とリード・レジスタ(RR0～RR6)からなります．図9にライト・レジスタの構成を示し，図10にリード・レジスタの構成を示します．

ライト・レジスタは7個あるというよりは，7グループあるというべきであり，各グループはまたいくつかのレジスタ群からなります（WR5を除く）．

Z80 DMAの動作は，このライト・レジスタに対するプログラミングにより決定されます．各ライト・レジスタのベース・レジスタは，ポインタ・ビットと呼



ばれるビットを持っており，ベース・レジスタに対するI/Oライト命令に続く，I/Oライト命令により書き込まれるべきレジスタを指定します．

リード・レジスタもやはり7個あります．これらにはDMA動作終了時のステータス，バイト・カウンタの値，A/B両ポートのアドレス・カウンタの値が保持されています．リード・レジスタの内容は常に読み取り可ですが，通常はDMA動作終了時に読み取りを行い，DMA動作の終了原因を判断したり，DMAデータ転送により転送された，データのバイト数を知るのに用いられます．これらのレジスタについては，「Z80 DMAのプログラミング」の項で詳しく述べます．

・パルス制御部

Z80 DMAは256バイトのDMAデータ転送をするごとに，$\overline{\text{INT}}$ライン上にパルスを発生する機能を持っています．また最初のパルス発生タイミングはライト・レジスタ4のパルス制御バイトの値だけオフセットすることができます．パルス制御部はこれらの制御を行います．

・一致検出ロジック

Z80 DMAはDMA転送中に転送されるデータを監視し，一致を検出すると割り込みを発生する機能を持っています．一致検出ロジックはこの制御を行います．

・バイト・カウンタ

これはDMAデータ転送開始時にクリヤされ，1バイトのデータを転送するごとに1だけ進められるカウンタです．このカウンタの値がライト・レジスタ中のバイト・カウント・レジスタの値と一致すると，DMAデータ転送は終了します．

・アドレス・カウンタ

Z80 DMAにはアドレス・カウンタとしてポートAアドレス・カウンタと，ポートBアドレス・カウンタがあり，両者ともアップ／ダウン・カウンタです．Z80 DMAでは，DMAデータ転送はポートAとポートB間で行われ，どちらがソース（読まれるほう）になっても，どちらがデスティネーション（書かれるほう）になってもかまいません．またポートA，ポートBはそれぞれ独立にメモリ，またはI/Oと指定することができます．

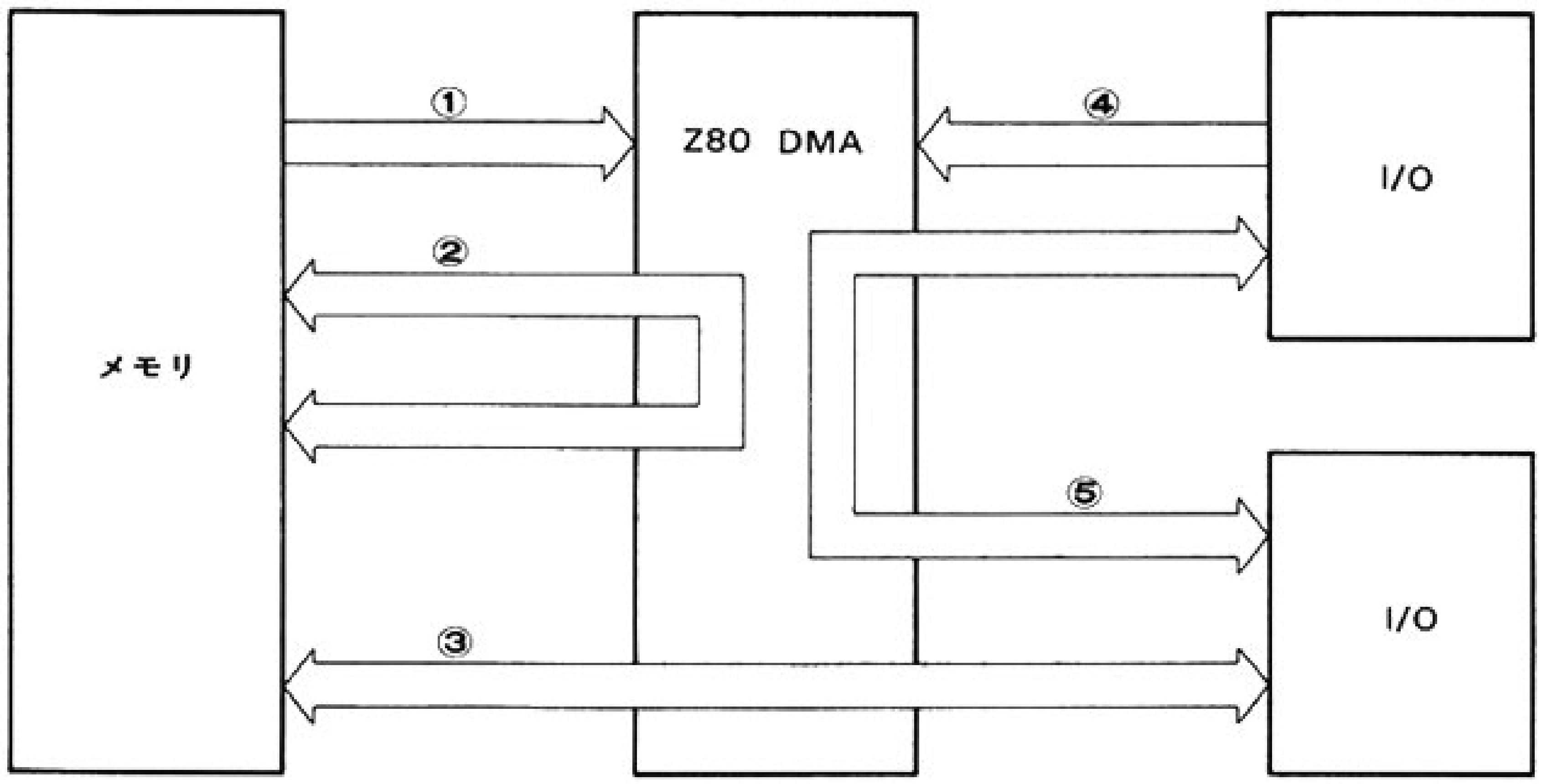
アドレス・カウンタはポートのアドレスを保持するカウンタであり，1バイトのDMAデータ転送が終わるごとに，1だけ進めたり減じたりすることができます．またこのカウンタの内容を変化させず，常に一定値をとらせる（Fixed Address）こともできます．

DMAデータ転送時にはこれらのアドレス・カウンタの状態が，アドレス・バス・バッファを介してシステム・バス上に出力され，メモリまたはI/Oをアドレッシングします．DMAデータ転送開始時には，ライト・

〈図10〉 Z80 DMAのコントロール・レジスタ（リード・レジスタ）の構成．リード・レジスタ0～6に対して，RR0～RR6の名前が付けられている



〈図11〉
Z80 DMA のデータ転送パス



① サーチ・メモリ
② メモリ↔メモリ*
③ メモリ↔I/O*
④ サーチI/O
⑤ I/O↔I/O*
(*印：サーチ併用可)

レジスタ0(WR0)のポートAスタート・アドレス・レジスタの内容をポートAアドレス・カウンタにロードし，ライト・レジスタ4 (WR4) のポートBスタート・アドレス・レジスタの内容を，ポートBアドレス・カウンタにロードします．

Z80 DMAの機能

■DMA データ転送パス (Pass)

図11に Z80 DMA が取り扱うことのできる，DMA データ転送パスを示します．この図のように Z80 DMA は5種類のデータ転送パスを取り扱うことができます．

① サーチ・メモリ
② メモリ ↔ メモリ (サーチ併用可)
③ メモリ ↔ I/O (サーチ併用可)
④ サーチ I/O
⑤ I/O ↔ I/O (サーチ併用可)

Z80 DMA がこのように多くのデータ転送パスのバリエーションを持っているのは，ポートAおよびポートBをそれぞれ独立にメモリ，またはI/Oと設定できることと，リード・データに対するサーチ機能があるためです．次に個々の DMA データ転送パスの応用を考えてみます．

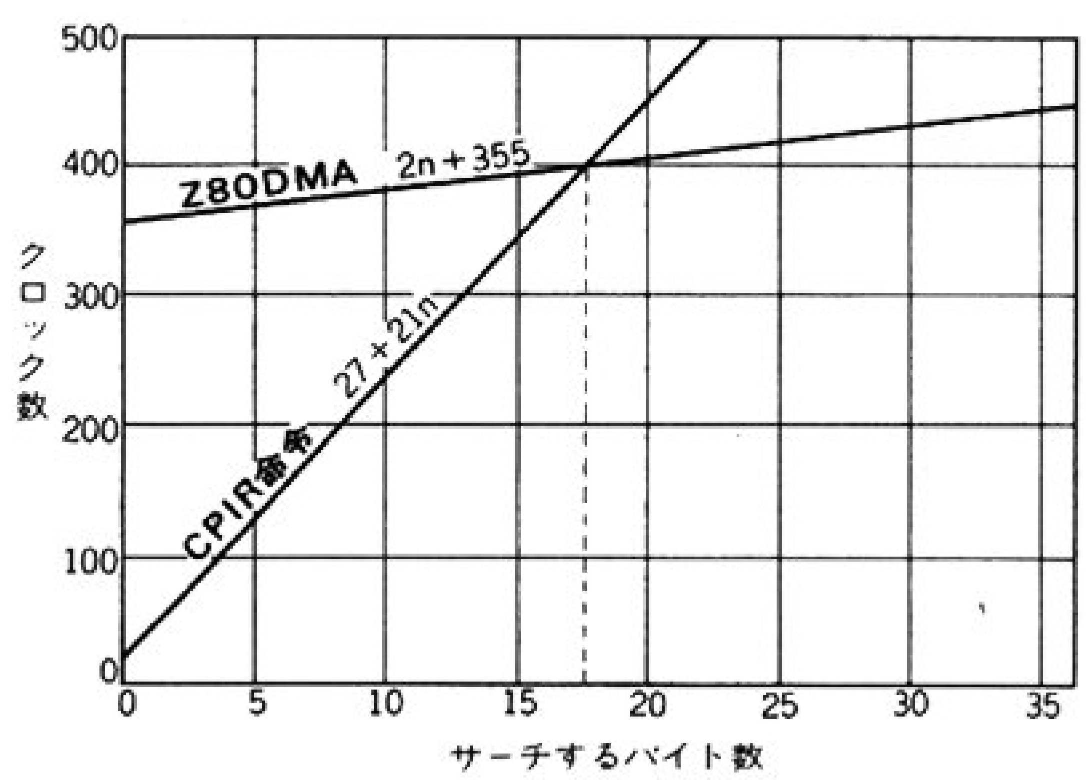
・サーチ・メモリ

Z80 CPUが持つ，CPIR命令と同じ機能を Z80 DMA が行います．したがって高速に多量のメモリ・データをサーチする場合に利用します．CPIR 命令では，1バイトのメモリ・サーチに要する時間は，21クロック期間ですが，Z80 DMA を用いると2クロック期間で，1バイトのサーチができます（2サイクル可変タイミングを用いた場合)．

図12に Z80 DMA のサーチ・メモリの機能を用いた場合と，Z80 CPU の CPIR 命令を用いた場合の，サーチすべきデータのバイト数と必要なクロック数との関係をグラフで示します．

Z80 DMA を用いた場合には，DMA のイニシャライズに 376 クロック程度を要しますので，サーチするデータのバイト数が少ないときには，CPIR 命令のほうが有利ですが，バイト数が19を越えると Z80 DMA のほうがだんぜん有利になります．

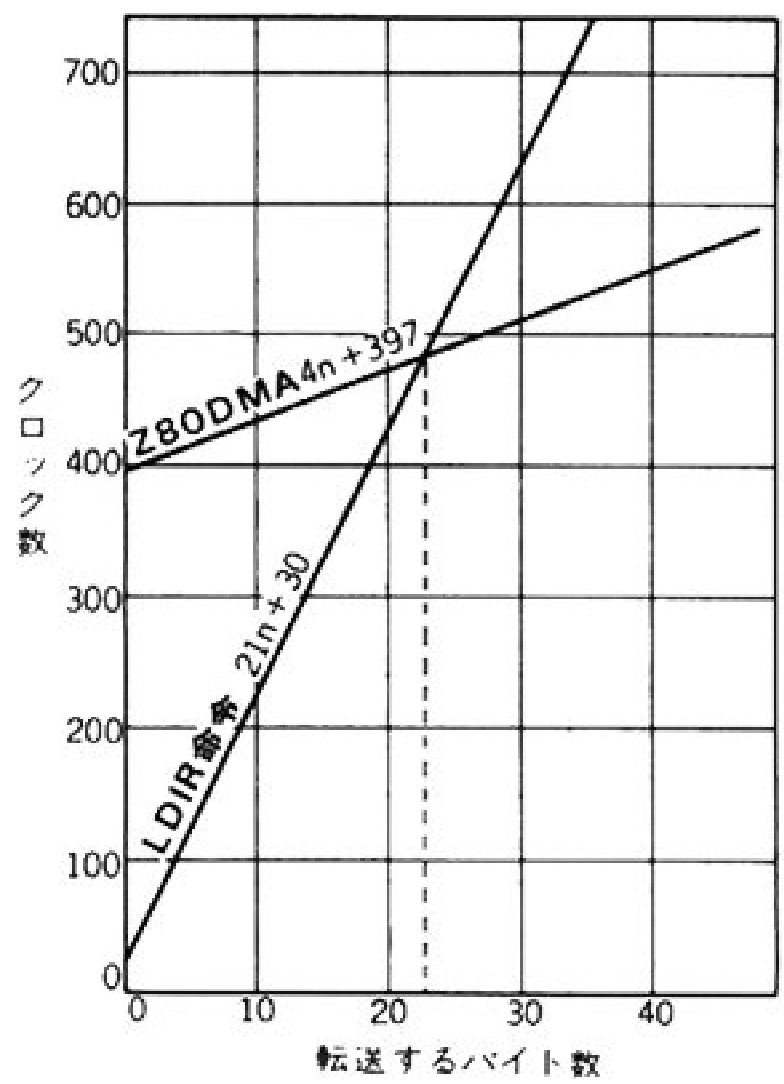
〈図12〉 メモリ・リサーチ.Z80 DMA を用いた場合と Z80 CPU の命令 CPIR を用いた場合の所要クロック数．19バイト以上なら DMA のほうが有利（2サイクル可変タイミングを用いたとき）



・メモリ ↔ メモリ

Z80 CPU の持つ LDIR 命令と同じ働きをします．したがって高速に多量のデータをメモリ間で転送する場合に利用します．LDIR 命令では1バイトの転送に21ステートを要しますが，Z80 DMA を用いると4ステートですみます（2サイクル可変タイミングを用いた場合)．

図13に LDIR 命令と，Z80 DMA を使用した場合の，転送すべきデータのバイト数と必要なクロック数の関係を示します．Z80 DMA を用いた場合には，DMA のイニシャライズに 420 クロック程度を必要としますの



〈図13〉
メモリ→メモリ間データ転送．Z80 DMAを用いた場合と，LDIR命令を用いた場合の所要クロック数．25バイト以上ならDMAのほうが有利
（2サイクル可変タイミングを用いたとき）

て，転送すべきバイト数が少ないときにはLDIR命令のほうが有利ですが，25バイトを境にしてDMAのほうが有利になることがわかります．

・メモリ↔I/O

これはメモリとI/O間のデータ転送であり，最も普通でかつまともなデータ転送パスです．一般的にいってもこの転送パスを取り扱うことができさえすれば，DMAコントローラとしての資格は与えられ，他の機能はおまけであるといっても過言ではありません．

このデータ転送パスの応用として，ディスケット（またはディスク）→メモリ，メモリ→ディスケット（またはディスク）の応用が代表的であり，他にメモリ→高速ライン・プリンタ，高速紙テープ・リーダ→メモリ，それに高速シリアル・インターフェース（Z80 SIOなど）とのデータのやりとりをあげることができます．

・サーチI/O

このデータ転送パスが利用されることは，ほとんどないと思いますが，しいていえば磁気テープなどのファイル・マークの検出に用いることができます．ファイルの区切りを示すファイル・マークとして，データ中には用いられないようなコードを割り当てておき，あるファイルを読み飛ばして，その次のファイルをアクセスしたいようなときに，このデータ転送パスが利用できます．

Z80 DMAをサーチI/Oのモードに設定して，磁気テープを駆動し，Z80 DMAにファイル・マークのサーチをやらせ，プログラムは“知らん顔”，Z80 DMAがファイル・マークを見つけると，そのことを割り込みにより知らせます．そこでやおらプログラムが動き出して，アクセスしたいファイルの中身を見にくるというやりかたも可能です．

・I/O↔I/O

このDMAデータ転送パスも特殊であり，応用例を捜すのに骨が折れますが，使いかたによっては効率のよいシステムを作れるかもしれません．

仮りに一つのI/Oがバッファ付きのディスケットであるとし，他のI/Oがライン・プリンタであるとします．そしてディスケットの内容を何の細工もせずに，ライン・プリンタに吐き出す応用があるとします（このような応用は，アセンブリ・リストのプリント時など，数多く見られます）．

このような場合，ふつうはまずディスケット→メモリのデータ転送を，プログラム・データ転送またはDMAデータ転送により行い，次にメモリ→ライン・プリンタのデータ転送を同様の方法で行いますが，ディスケットをDMAデータ転送のソースとし，ライン・プリンタをデスティネーションとして，I/O→I/OのDMAデータ転送を行うことにより，プログラムはDMAデータ転送の起動だけを行い，あとはI/O同士が勝手にデータのやりとりを行う，というような応用も考えることができます．

これによりI/O同士がデータの授受をしている間，プログラムは他の仕事をやっていられますので，システムの効率を上げることができます．ただしこのような応用を行った場合，プログラムはI/O同士のやりとりに関知しないので，I/Oエラーが生じた場合，I/O同士でそのリカバリを行う必要があります．したがって，I/Oは互いに相当のインテリジェンシイを持つ必要があります．

・同時データ転送（Simultaneous Data Transfer）

以上に述べた五つのデータ転送パスのほかに，Z80 DMAでは「同時データ転送」という概念があります．これはメモリとI/Oとのデータ転送を高速化するためのテクニックであり，一種のごまかしであるということができます．

同時データ転送では，メモリとI/O間のデータ転送を行うときに，Z80 DMAにはメモリ・サーチ，またはI/Oサーチ動作を行わせておき，メモリ・リード，またはI/Oリードのタイミングに，外部ロジックにより，I/Oライト信号（$\overline{IORQ}$, $\overline{WR}$）またはメモリ・ライト信号（$\overline{MREQ}$, $\overline{WR}$）を発生させます．

Z80 DMAにメモリ・サーチをやらせておいて，メモリ→I/Oのデータ転送を行うときには，メモリ・リードのタイミングに，外部回路によりI/Oライト信

号を発生します.

また, I/O → メモリのデータ転送を行うときには, メモリ・リードのタイミングに, I/O リード信号($\overline{IORQ}$, $\overline{RD}$), およびメモリ・ライト信号を外部回路が発生します.

どちらの場合においても, Z80 DMA は一つのアドレスしか出力しませんので, これをメモリ・アドレスとして用いるとすると, I/O アドレスはありませんので, I/O のチップ・セレクトも外部回路で行います.

Z80 DMA に I/O サーチをやらせておいて, メモリ → I/O のデータ転送を行うときには, I/O リードのタイミングに, メモリ・リード信号 ($\overline{MREQ}$, $\overline{RD}$) および I/O ライト信号を外部回路により発生します.

I/O → メモリのデータ転送を行う場合には, I/O リードのタイミングに, メモリ・ライト信号を外部回路により発生します. これらの場合には, Z80 DMA は自分が出力するアドレスは I/O アドレスであると思っていますが, 外部ロジックによりこれをメモリ・アドレスとして取り扱います. したがって, この場合にも I/O のチップ・セレクトは外部回路により行います.

Z80 DMA が出力するアドレスを I/O アドレスとして扱わず, メモリ・アドレスとして扱う理由は, DMA データ転送中にメモリ・アドレスはほとんどの場合変化しますが(インクリメント/デクリメント), I/O アドレスは固定であることが多いので, I/O のセレクトは外部ロジックでも十分であるからです.

Z80 DMA は標準タイミングの場合, メモリ・リードは 3 クロック, I/O リードは 4 クロック期間で行いますので, この同時データ転送テクニックを用いれば, メモリ・サーチ・タイミングを利用したときには 3 クロック, I/O サーチ・タイミングを利用したときには 4 クロック期間でメモリと I/O 間のデータ転送ができます.

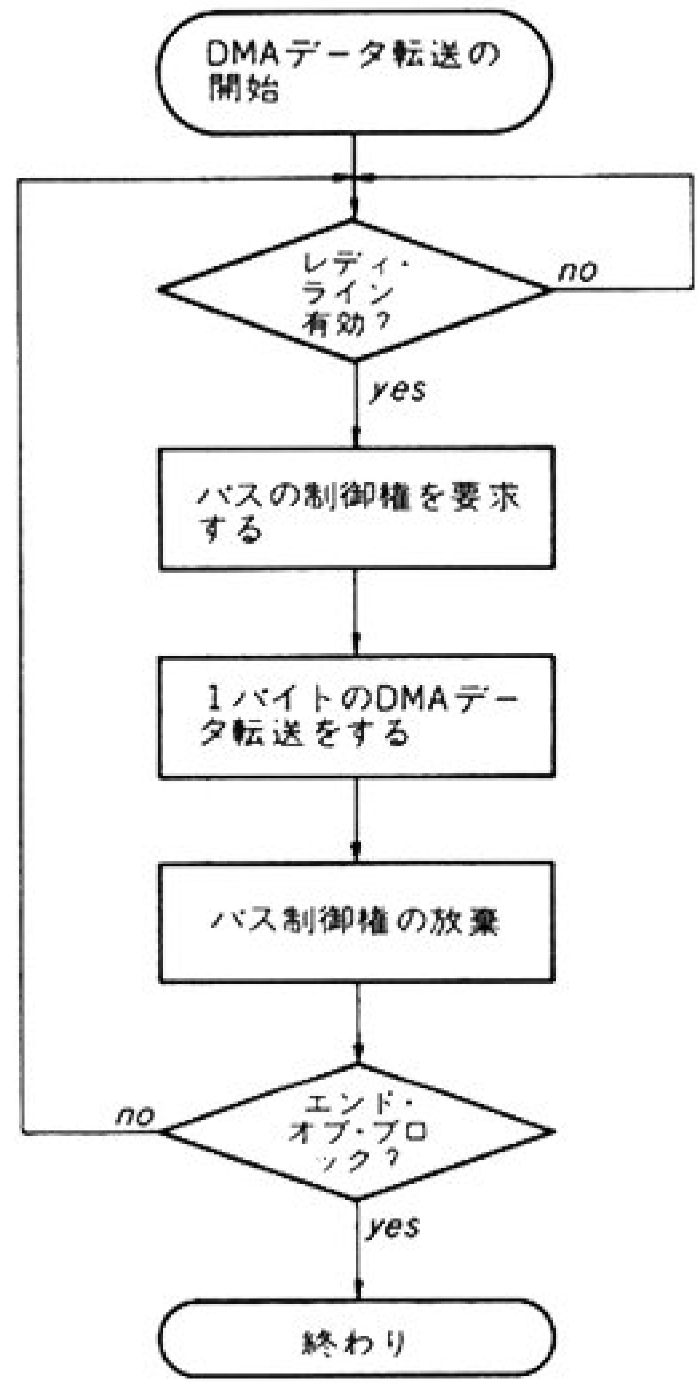
また Z80 DMA の可変タイミング機能を用いて, リード・サイクルを 2 クロック期間と設定することにより, メモリと I/O 間のデータ転送を 2 クロック期間で行うことができます. Z80 DMA のデータ転送速度の最大値, 2M バイト/秒 (クロック = 4 MHz) は, この同時データ転送を用い, かつリード・サイクルを 2 クロックに設定したときに得られます.

なお, この同時データ転送に対して, 通常のデータ転送のことを, 逐時(ちくじ)データ転送(Sequencial Data Transfer)と呼びます.

■Z80 DMA のデータ転送モード

Z80 DMA は 3 種類のデータ転送モードを持っています.

① バイト・モード (Byte)



〈図14〉 バイト・モードの DMA データ転送のフロー

② バースト・モード (Burst)

③ コンティニュアス・モード (Continuous)

以下にこれら三つの転送モードについて説明をします.

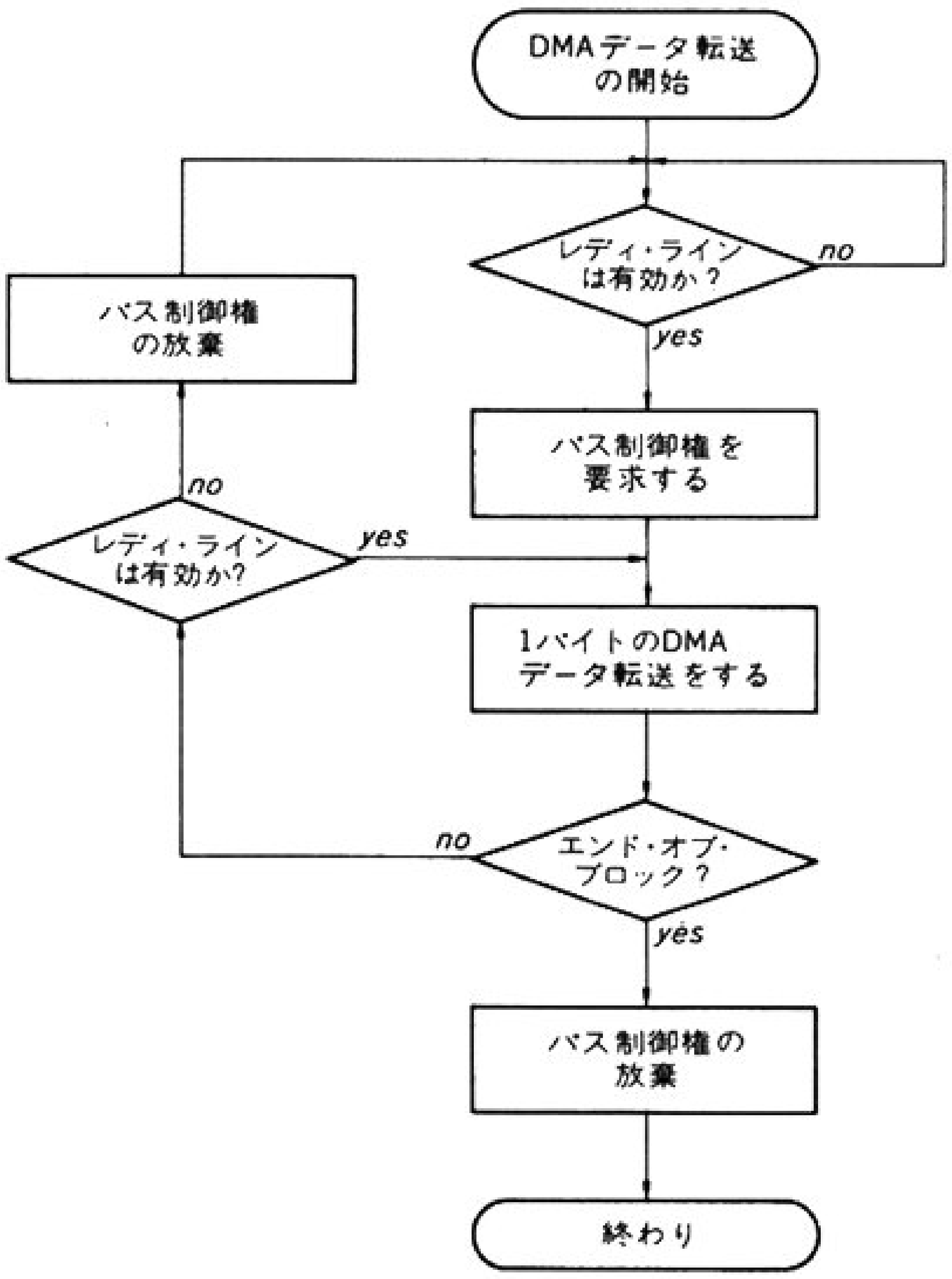
・バイト・モード

このモードでは, 1 バイトの DMA データ転送を行うごとに, バスの制御権を放棄し, 最低 1 マシン・サイクル期間, バスの制御権を CPU に返します. バスの制御権を CPU に返してから, 1 マシン・サイクル経(た)ったときに, Z80 DMA のレディ・ラインの状態が, 有効レベルになっているときには, 再度 CPU に対してバスの制御権を要求し, 次の 1 バイトのデータ転送をします.

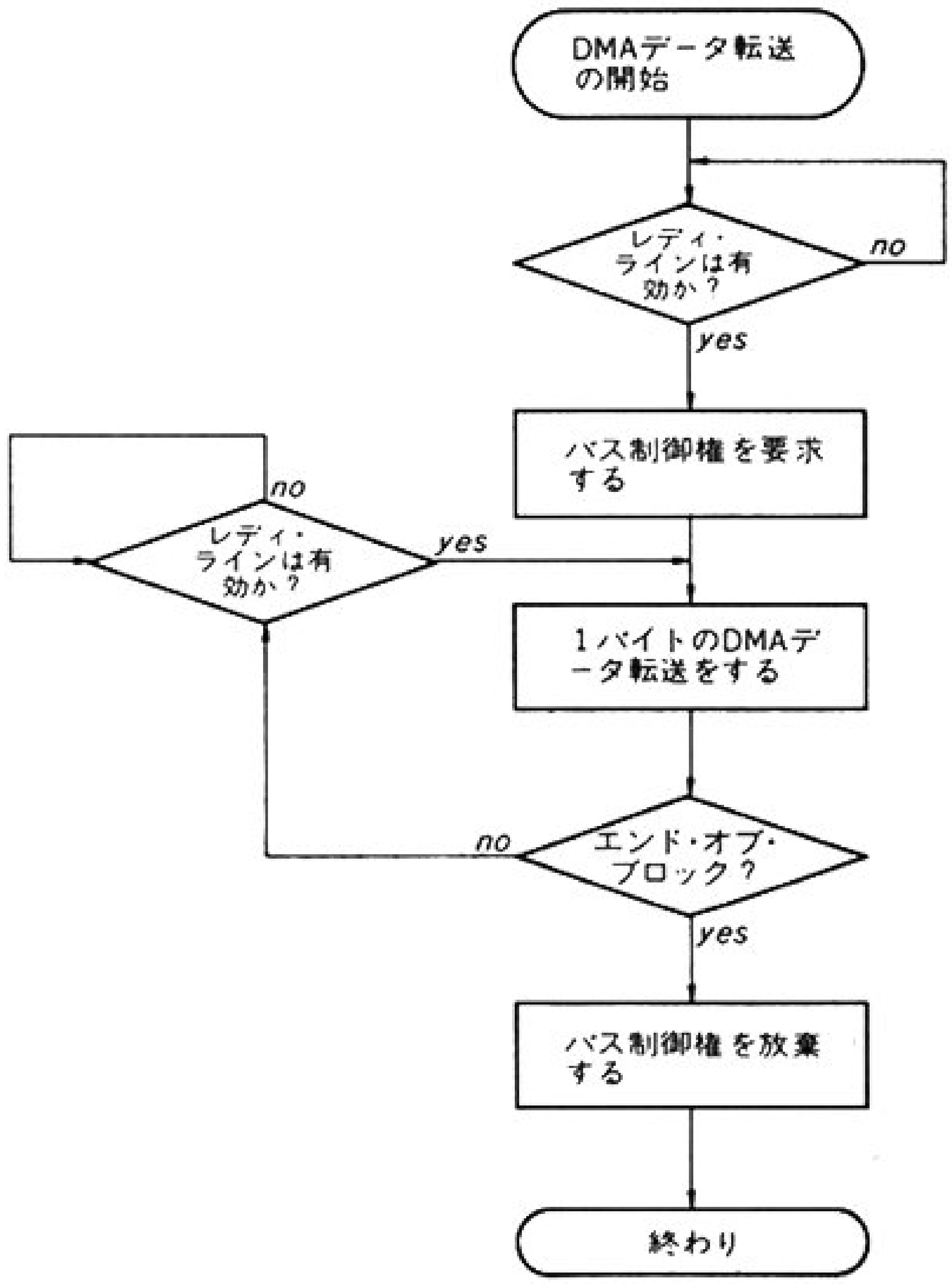
バスの制御権を CPU に返してから, 1 マシン・サイクル経った時点に, レディ・ラインの状態が有効レベルにないときは, バスの制御権放棄を継続します. 図14にバイト・モードの DMA データ転送のフローを示します.

このモードは I/O のデータ転送速度が比較的遅く, DMA データ転送をしていないときにも, CPU を止めておくのはもったいないというような場合や, 1 ブロックの長さが非常に長いため, 1 ブロックの DMA データ転送に相当の時間がかかり, その期間中 CPU を

〈図15〉 バースト・モードのDMAデータ転送フロー



〈図16〉 コンティニュアス・モードのDMAデータ転送のフロー



止めておいたのでは，他の処理（たとえばタイマ割り込み）に支障をおよぼすような場合に用います．

・バースト・モード

図15にバースト・モードのDMAデータ転送のフローを示します．このモードと前のバイト・モードとの相違点は，1バイトのDMAデータ転送をした後に，無条件にバス制御権を放棄するのではなく，レディ・ラインの状態を調べ，それが有効レベルにあれば，引き続き次のデータを転送し，レディ・ラインの状態が有効レベルでなくなったときに，はじめてバス制御権を放棄する点にあります．

したがってこのモードでは，I/Oがデータ転送をしたいといっている期間中（レディ・ラインが有効レベル），たて続けにデータ転送をすることができ，I/Oがデータ転送をしなくてもよいと思っている期間（レディ・ラインが無効レベル）はCPUが走ることができますので，データ転送速度およびバスの使用効率の面からみても理想的なモードであるということができます．

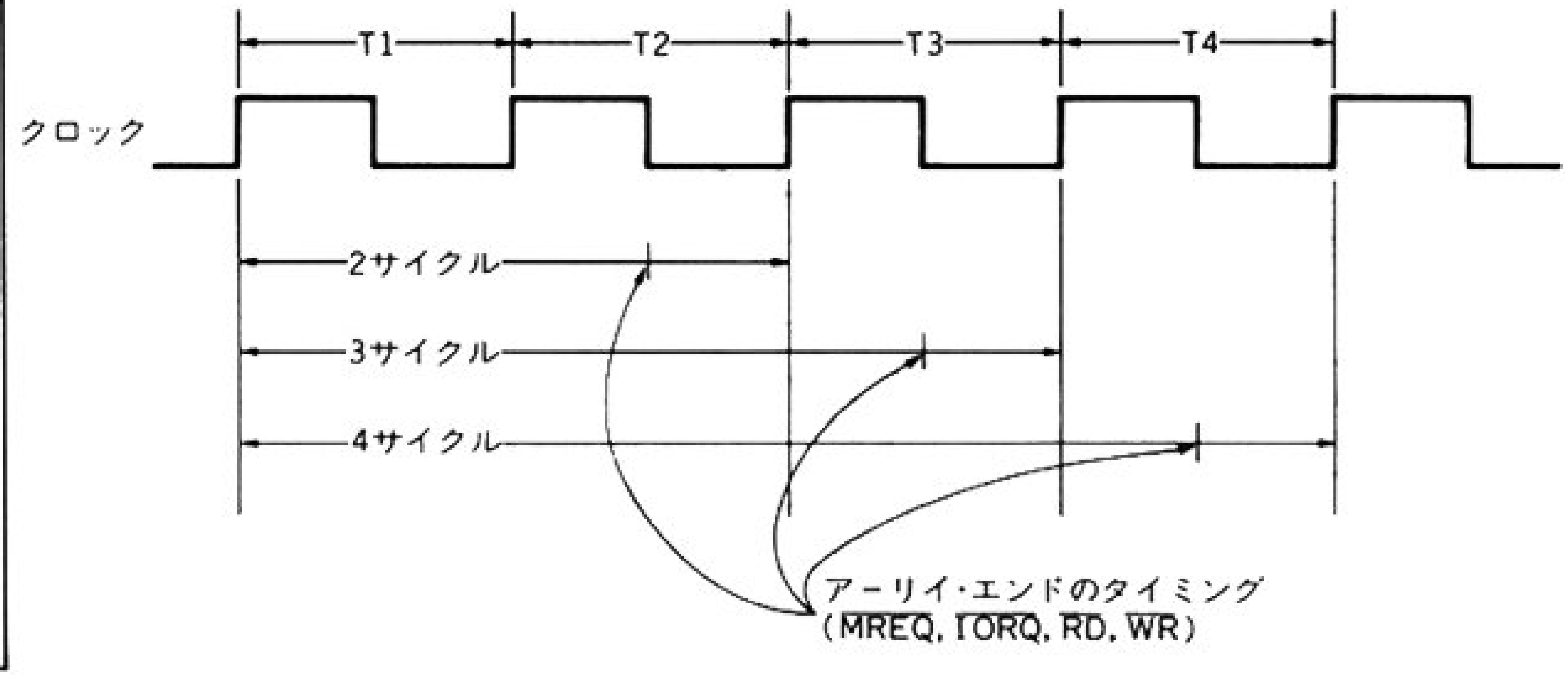
・コンティニュアス・モード

図16にコンティニュアス・モードのDMAデータ転送のフローを示します．このモードではいったんDMAデータ転送が開始されると，エンド・オブ・ブロックになる（バイトカウンタ＝バイト・カウント・レジスタ）まで，Z80 DMAはバスの制御権を放棄しません．1バイトのDMAデータ転送をするごとにレディ・ラインの状態を調べますが，たとえレディ・ラインの状態が無効レベルにあっても，DMAデータ転送を行わないだけであり，バスの制御権を握ったままです．

このモードはI/Oのデータ転送速度が非常に高速であり，かつ転送すべきデータのバイト数があらかじめわかっているようなときに用います．もしI/Oとやりとりするデータのバイト数が，Z80 DMAのバイト・カウンタにセットした値よりも小さいと，永久にブロック終了の状態にならず，バスの制御権はZ80 DMAに渡されたままになり，システムはハング・アップします．

■可変タイミング機能

Z80 DMAがDMAデータ転送時に実行する，メモリやI/Oに対するリード／ライト・サイクルは，デフォールト時（プログラムにより変更しないとき）には，Z80 CPUのタイミングと同一です．したがって，メモリに対するリード／ライト・サイクルは3クロック期間であり，I/Oに対するリード／ライト・サイクルは4クロック期間です．



〈図17〉
可変タイミング

ところが，Z80 DMA には“可変タイミング”という機能があり，プログラムにより上記のサイクルのクロック数を2～4の範囲で変更することができます．またこの変更はポートA，ポートBに対して独立に行うことができます．

さらにこれらのサイクルで用いられる四つの信号，$\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, $\overline{WR}$ の立ち上がりタイミングを½クロックだけ早くする(アーリイ・エンド：Early End)機能もあります．図17にこの可変タイミング機能を示します．

この可変タイミングは，Z80 CPU のタイミング(標準タイミングと呼ぶ)では遅過ぎてもったいないような高速のメモリやI/O間のデータ転送を，Z80 DMA を用いて行うようなときに，転送速度を上げるために利用します．

私たちがふつう使用しているメモリはかなり高速ですので，メモリ・リード／ライト・サイクルは2クロックで十分です．したがって，この機能を用いて，メモリと指定されているポートのタイミングを2クロックとすることにより，メモリとのやりとりを行う速度は1.5倍になります．

ここで注意すべきことは，Z80ファミリの周辺 LSI (たとえばZ80 PIO, Z80 SIO) を DMA データ転送の対象としているときに，この可変サイクルを用いると，それらの LSI の動作は保証されません．Z80ファミリの LSI は Z80 CPU の標準タイミングで動作するように設計されているからです．ただし，アタックしてみる価値はあるかもしれません．

■オート・リスタート機能

通常の DMA データ転送は，エンド・オブ・ブロックになると終了しますが，オート・リスタート機能(プログラムにより指定できる) を用いると，エンド・オブ・ブロック時に，自動的にアドレス・カウンタにライト・レジスタ中のスタート・アドレス・レジスタの内容がロードされ，バイト・カウンタはゼロ・クリヤされ，DMA データ転送を再開することができます．

この機能は，CRT ディスプレイのスクリーン・リフレッシュなどの，周期的にある一定のメモリ・ブロック (スクリーン・メモリ) をアクセスする必要のある場合に有効です．

■Z80 DMA の割り込み

Z80 DMA は次に示す三つの状態のときに割り込みを発生することができます．

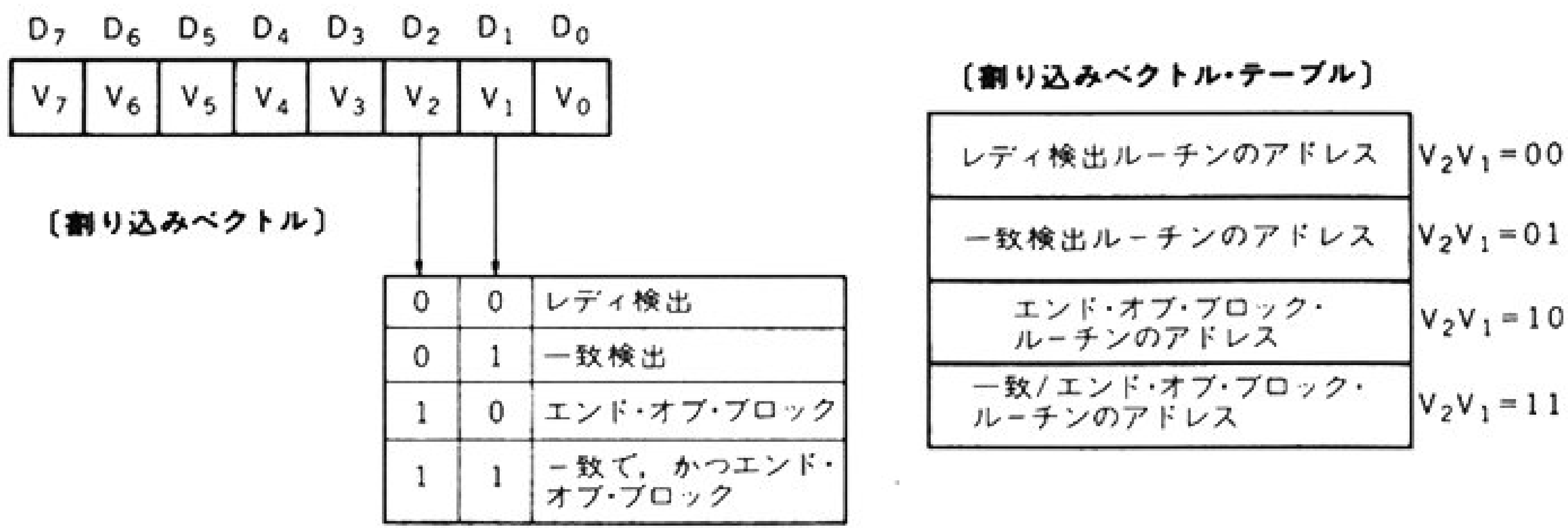
① レディ検出 (Interrupt on Ready)
② 一致検出 (Interrupt on Match)
③ エンド・オブ・ブロック時 (Interrupt on End of Block)

割り込み制御バイトをプログラミングすることにより，これら三つの状態のいずれかが生じたときに，割り込みを発生させることもできますが，割り込みを必要としない状態に対してマスクをかけることもできます．

Z80 DMA は，Z80 ファミリのデイジー・チェイン式，優先割り込み機能を持っており，割り込み時にはあらかじめプログラムされた，割り込みベクトルを発生します．ステータス・アフェクツ・ベクトル (Status Affects Vector) 機能を用いない場合には，上記三つの状態の中の，いずれが生じても同一の割り込みベクトルを発生しますので，割り込みの要因がいくつかあるような応用では，Z80 DMA のステータス・バイト (RR0) を読み込んで，割り込み原因を識別する必要があります．

これに対して，ステータス・アフェクツ・ベクトル機能を用いると，Z80 DMA は割り込みベクトルのビット1，およびビット2を，割り込み要因に応じて変化させてくれますので，割り込みルーチンがステータス・バイトを読んでチェックする作業は不要になります．

図18にステータス・アフェクツ・ベクトル時の割り込みベクトル，および割り込みベクトル・テーブルを



〈図18〉 ステータス・アフェクツ・ベクトル時の割り込みベクトルと割り込みベクトル・テーブル

示します．

■パルス発生機能

Z80 DMAは，256バイトのDMAデータ転送を行うごとに，$\overline{INT}$ ライン上にパルスを発生する機能を持っています．パルスを $\overline{INT}$ ライン上に出しますので，CPUがふつうの割り込み要求と間違えて解釈しないよう，このパルスはCPUの $\overline{BUSACK}$ 信号が"L"のときに出力されます．したがって割り込み機能とパルス発生機能を同時に用いるときには，$\overline{INT}$ と $\overline{BUSACK}$ をANDゲート（ローAND）を介してパルス出力を抽出する必要があります．

このパルス発生機能の応用としては，フロッピー・ディスク・コントローラ用LSI（たとえばμPD765）に対する，TC（ターミナル・カウント）入力信号を作る場合をあげることができます．

〈図19〉 Z80 DMA/Z80A DMAのデータ転送速度

データ転送形式	タイミング	データ転送速度	
		Z80 DMA	Z80A DMA
同時データ転送	標準	メモリ→I/O:0.833 I/O→メモリ:0.625	メモリ→I/O:1.33 I/O→メモリ:1.00
同時データ転送	2サイクル	1.25	2.00
逐時データ転送	標準	0.357	0.571
逐時データ転送	2サイクル	0.625	1.00

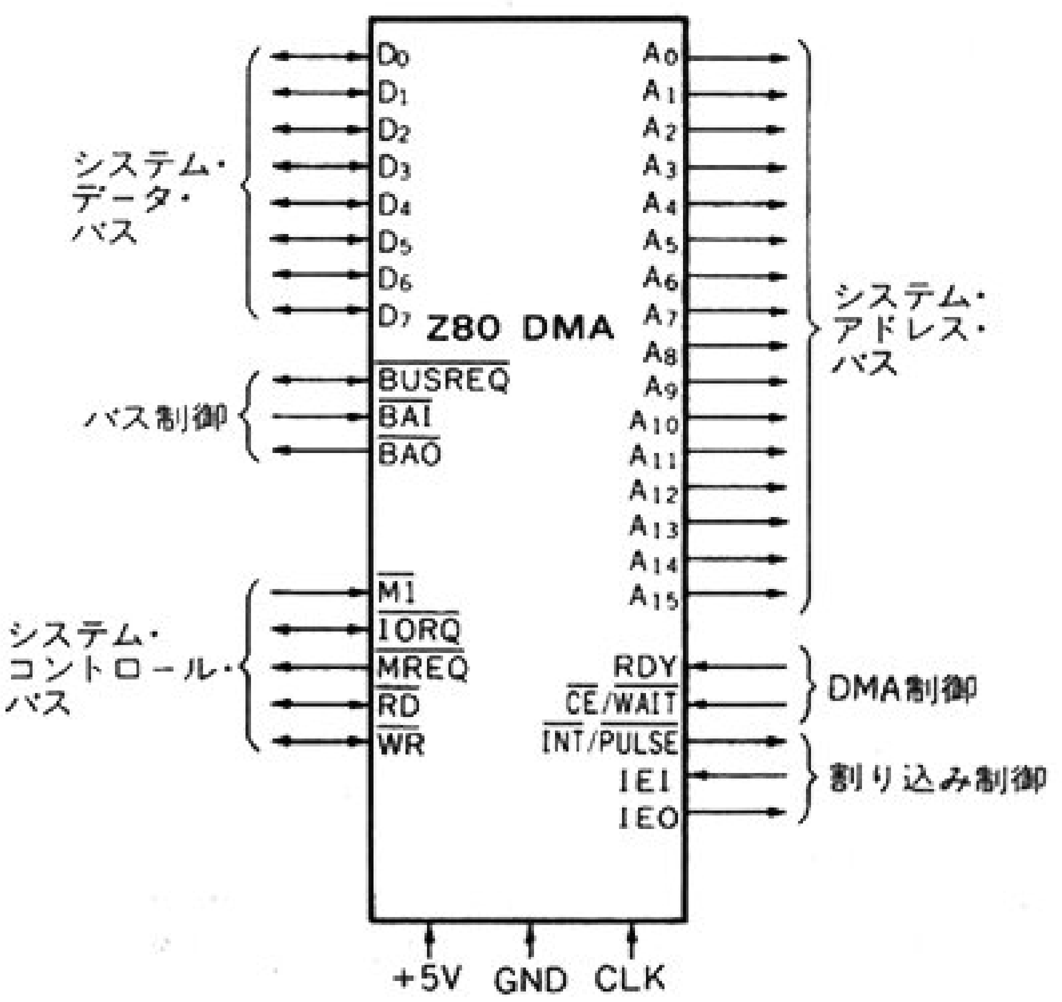
（単位：Mバイト/秒）

Z80 DMAのデータ転送速度

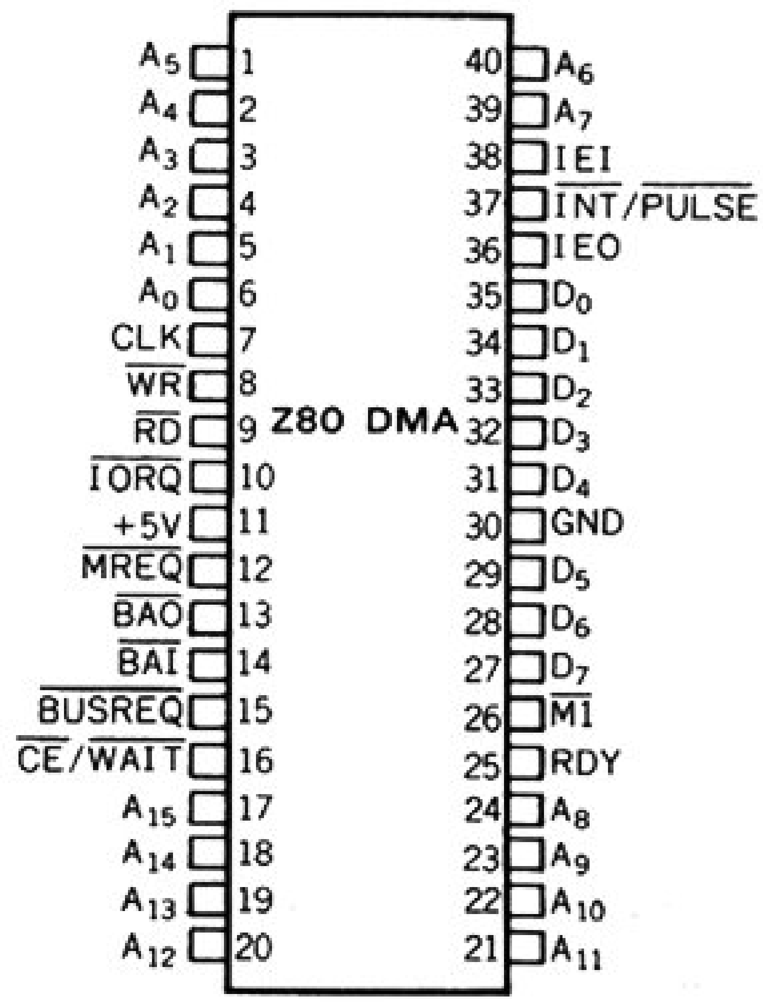
図19にZ80 DMAとZ80A DMAのデータ転送速度を示します．これまでの説明では一概に，Z80 DMAと呼んできましたが，Z80ファミリのDMAには，CPUやその他のLSIと同様に，Aバージョンとノン Aバージョンがあります．

すなわち，Aバージョンは最高クロック速度＝4MHzで動作し，ノンAバージョンは2.5MHzで動作します．最近Bバージョン（6MHz）も出回わってきましたが，Z80B DMAのデータ・シートを入手しておりませんので，ここではとりあげないことにします．

〈図20〉 Z80 DMAのピン機能



〈図21〉 Z80 DMAのピン配置



Z80 DMAのピン機能

Z80 DMAは，Z80 CPU，-SIO，-PIOなどと同じく，40ピン DIPに収められています．図20にZ80 DMAのピンを機能別に分けた図を，図21にチップの各ピンの配置を示します．また，図22に各ピンの機能をまとめた表を示します．以下に各ピンの機能について詳細に説明をします．

・A_0〜A_{15}(システム・アドレス・バス——出力，3ス

〈図22〉 Z80 DMAのピン機能

ピン名	信号名	I/O	出力形式	有効レベル	機能
A_0〜A_{15}	システム・アドレス・バス	出力	3ステート		DMAデータ転送時に出力バッファがイネーブルされ，ソース・アドレスおよびデスティネーション・アドレスが出力される
$\overline{BAI}$	バス・アクノウリッジ・イン	入力		L	この信号が"L"であると，Z80 DMAにバスの制御権があることを示す．この信号と$\overline{BAO}$とにより，バス・リクエスト・デイジー・チェインを形成する
$\overline{BAO}$	バス・アクノウリッジ・アウト	出力	TTL	L	この信号は，当該Z80 DMAがバスを使用しているときに"H"となる．またバスを使用しておらず，かつ$\overline{BAI}$が"L"のときには，この信号は"L"になる．$\overline{BAI}$とともにバス・リクエスト・デイジー・チェインを形成し，より優先度の低いZ80 DMAの$\overline{BAI}$と接続される
$\overline{BUSREQ}$	バス・リクエスト	双方向性	オープン・ドレイン	L	1.8kΩ程度の抵抗により，プルアップしてCPUの$\overline{BUSREQ}$ピンに接続する．複数のZ80 DMAを用いるときには，ワイヤードOR接続する．この信号はCPUに対するバス・リクエスト信号であるとともに，他のZ80 DMAがバス・リクエストを行っているか否かの判断に用いられる
$\overline{CE}$/$\overline{WAIT}$	チップ・イネーブル/ウェイト	入力		L	通常はチップ・イネーブル($\overline{CE}$)の機能として動作し，アドレス・デコーダの出力が接続されるが，プログラムにより$\overline{WAIT}$機能を用いるようにすると，DMAデータ転送時にはウェイト信号として動作をする
CLK	システム・クロック	入力			Z80 DMAに対するクロック入力であり，Z80 CPUのクロックと同じものが使用できる．クロック周波数の最大値は，Z80 DMAは2.5MHz，Z80A DMAでは4MHzである
D_0〜D_7	システム・データ・バス	双方向性	3ステート		DMAデータ転送をしていないときには，Z80 DMAのプログラミングおよびステータスの読み込みにこのデータ・バスが使われる．DMAデータ転送中には，ソース・ポートのデータがいったんZ80 DMA内に入り，それからデスティネーション・ポートに書き込まれるが，このときにもこのデータ・バスが用いられる
IEI	インタラプト・イネーブル・イン	入力		H	この信号が"H"であると，当該Z80 DMAより割り込み優先度の高いペリフェラルが，割り込み要求を出していないことを示す
IEO	インタラプト・イネーブル・アウト	出力	TTL	H	この信号が"H"であると，当該Z80 DMAおよびそれより割り込み優先度の高いペリフェラルが割り込み要求を出していないことを示す． IEIとともに割り込みデイジー・チェインを形成する
$\overline{INT}$/$\overline{PULSE}$	割り込み要求/パルス出力	出力	オープン・ドレイン	L	CPUに対する割り込み要求線($\overline{BUSACK}$="H")． $\overline{BUSACK}$="L"のときには，256バイトのDMAデータ転送を行うごとに，パルスが出力される
$\overline{IORQ}$	I/Oリクエスト	双方向性	3ステート	L	DMAデータ転送をしていないときには($\overline{BUSACK}$="H")，入力となり，CPUの$\overline{IORQ}$を受け付ける．DMAデータ転送時($\overline{BUSACK}$="L")には出力となり，I/Oのデータを読み書きするときに，Z80 DMAがこの信号を制御する
$\overline{M1}$	マシン・サイクル1	入力		L	Z80 CPUの$\overline{M1}$信号がそのまま接続される．Z80 DMAはこの信号を割り込み認識シーケンスの識別に用いる
$\overline{MREQ}$	メモリ・リクエスト	出力	3ステート	L	DMAデータ転送を行うときに，出力がイネーブルされ，メモリのデータを読み書きするときに，Z80 DMAはこの信号を制御する
$\overline{RD}$	リード	双方向性	3ステート	L	DMAデータ転送をしていないときには入力となり，CPUの$\overline{RD}$信号を受け付ける．DMAデータ転送時には出力となり，メモリやI/Oのデータを読むときに，Z80 DMAがこの信号を制御する
RDY	レディ	入力		プログラム可	この信号が有効レベルにあると，Z80 DMAはDMAデータ転送を行う．有効レベル(H/L)はプログラムにより指定する
$\overline{WR}$	ライト	双方向性	3ステート	L	DMAデータ転送をしていないときには入力となり，CPUの$\overline{WR}$信号を受け付ける．DMAデータ転送時には出力となり，メモリやI/Oにデータを書き込むときに，Z80 DMAがこの信号を制御する

テート）

これらのラインは，DMA データ転送時に"出力"としてイネーブルされ，ソース・アドレス，およびデスティネーション・アドレスが出力されます．これらのライン上のアドレスは，メモリ・アドレスであることもありますし，I/O アドレスであることもあります．

・$\overline{\text{BAI}}$（バス・アクノウリッジ・イン——入力，"L"で有効）

BAI は Bus Acknowledge In の略です．このラインの状態が"L"であると，当該 Z80 DMA がシステム・バスを使用する権利があることを示します．

このラインは，システム中に Z80 DMA が一つしかない場合，およびシステム中に Z80 DMA が複数あるが，当該 Z80 DMA が最高のバス使用優先順位を持っている場合には，Z80 CPU の $\overline{\text{BUSACK}}$ ピンに接続されます．また，システム中に複数個の Z80 DMA が用いられており，かつ当該 Z80 DMA のバス使用優先順位が最高位にない場合には，それよりも優先順位の高い Z80 DMA の $\overline{\text{BAO}}$ に接続されます．

・$\overline{\text{BAO}}$（バス・アクノウリッジ・アウト——出力，"L"で有効）

BAO は Bus Acknowledge Out の略です．このピンの信号は，システム内に複数の Z80 DMA を使用するときに，$\overline{\text{BAI}}$ とともに用いてバス使用権の優先順位制御を行います．したがって，Z80 DMA を一つしか使わないシステムでは，この信号は不用です．

この信号はシステム中に複数の Z80 DMA が用いられているときに，自分よりバス使用優先順位の低い Z80 DMA の $\overline{\text{BAI}}$ ピンに接続され，いくつかの Z80 DMA が同時に CPU に対してバス・リクエストを出した場合に，バス・リクエストを出した Z80 DMA のうちの，最高優先順位のものが $\overline{\text{BAI}}$ を獲得し，自分の $\overline{\text{BAO}}$ を"H"にすることにより，優先度の低い Z80 DMA がバスを使用することを抑えます．

もちろん，バス・リクエストを行っていない Z80 DMA は，$\overline{\text{BAI}}$ の状態をそのまま $\overline{\text{BAO}}$ にパスします．

・$\overline{\text{BUSREQ}}$（バス・リクエスト——双方向性，"L"で有効，オープン・ドレイン型出力）

この信号は双方向性であり，"出力"の場合には CPU に対するバス・リクエスト信号として用いられ，DMA データ転送に先立って，バスの制御権を CPU から譲り受けるのに使われます．

"入力"信号として用いられるときには，バス上にぶら下がっている他の Z80 DMA が，バス・リクエストを出していないかどうかを，チェックするのに使われます．

この信号の出力はオープン・ドレイン型ですので，1.8kΩ 程度の抵抗によりプルアップして，CPU の $\overline{\text{BUSREQ}}$ ピンに接続します．複数の Z80 DMA を用いるときには，各 DMA の $\overline{\text{BUSREQ}}$ 信号をワイヤード OR 接続して，CPU の $\overline{\text{BUSREQ}}$ ピンに接続します．

Z80 DMA では複数の DMA を同一バス上で用いるときには，図23に示すバス・リクエスト・デイジー・チェインにより，バス使用優先順位の制御を行います．各 DMA の $\overline{\text{BUSREQ}}$ 信号は，ワイヤード OR 接続されて，CPU の $\overline{\text{BUSREQ}}$ ピンに接続されています．

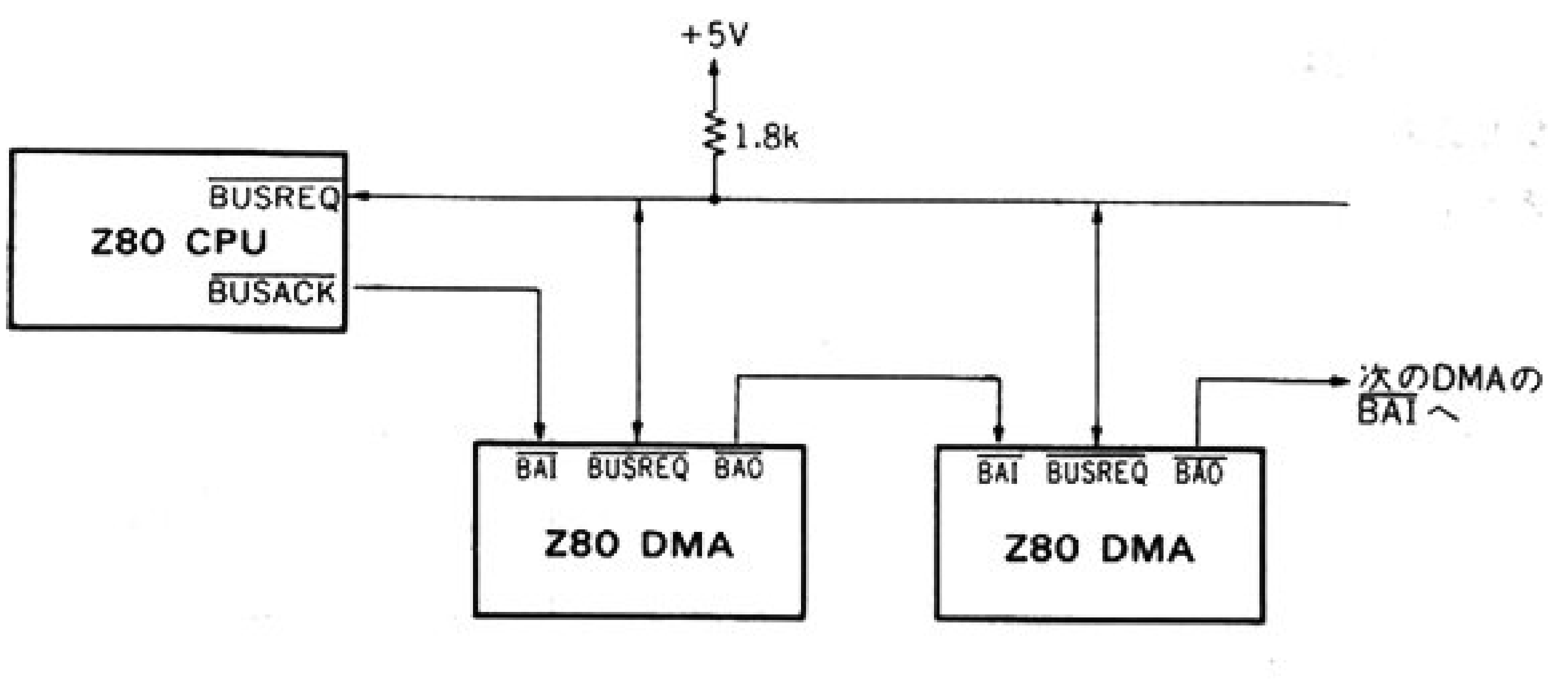
CPU の $\overline{\text{BUSACK}}$ 信号は，左端の DMA の $\overline{\text{BAI}}$ に接続され，この DMA の $\overline{\text{BAO}}$ は次の DMA の $\overline{\text{BAI}}$ に接続されています．バス使用優先順位は，$\overline{\text{BAI}}$ 信号が CPU の $\overline{\text{BUSACK}}$ 信号に近いものほど高く，遠いものは低くなります．図23は左端の DMA がもっとも優先度が高く，右端の DMA はもっとも低くなります．

このバス・リクエスト・デイジー・チェインは，Z80システムの割り込みデイジー・チェインによく似ていますが，バス・リクエストを先に出していれば，たとえ優先順位の低い DMA であっても，他の優先順位の高い DMA に，リクエストを横取りされることはないという点が異なります．

各 DMA はバス・リクエストを出す前に，自分の $\overline{\text{BUSREQ}}$ ピンを入力として，他の DMA がバス・リクエストを出していないことを確認したうえで，自分のバス・リクエストを出すようになっています．

したがって，このバス・リクエスト・デイジー・チェインは，互いに他の DMA がバス・リクエストを出していることを識別できないようなタイミングで，複数の DMA がバス・リクエストを出したようなときに，バス・リクエストを出している DMA の中で最高順位

〈図23〉 Z80 DMA のバス・リクエスト・デイジー・チェイン



のDMAだけが，CPUの$\overline{BUSACK}$を獲得し，そのDMAは自分の$\overline{BAO}$を“H”にして，より優先度の低いDMAにはバスの制御権を渡さないためにあります．

・$\overline{CE}/\overline{WAIT}$（入力，“L”で有効）

このピンは$\overline{CE}$(チップ・イネーブル)，および$\overline{WAIT}$（ウェイト）の2役をします．通常はこのピンは$\overline{CE}$として働きますが，プログラミングによりDMAデータ転送時には$\overline{WAIT}$の機能を果たすことができます．

$\overline{CE}$としての機能はふつうのI/O（Z80 SIO,-PIO）の場合と同じで，この信号が“L”であり，かつ$\overline{IORQ}$が“L”のときに，当該Z80 DMAがセレクトされ，DMAのプログラミングを行ったり，ステータスを読んだりするのに用いられます．

プログラミングにより，このピンを$\overline{WAIT}$として用いるように設定すると，DMAデータ転送時($\overline{BUSACK}$=“L”)には，このピン上の状態を$\overline{WAIT}$信号と解釈し，Z80 DMAはI/Oやメモリとのデータ転送タイミング中にウェイト・タイミングを挿入します．

このように$\overline{CE}/\overline{WAIT}$ピンは2通りの意味を持ち得ますので，$\overline{WAIT}$機能を用いるときには，このピンに対する入力信号をマルチプレックスする必要があります．

・CLK（クロック，入力）

Z80 DMAの内部動作のために必要なクロックであり，CPUのクロックをそのまま用いることができます．このクロックの最高周波数はZ80 DMAでは2.5MHz，Z80A DMAでは4MHzです．

・D_0～D_7（システム・データ・バス――双方向性）

これらのピンは，Z80 DMAがDMAデータ転送をしていないときには，Z80 DMAのプログラミングおよびステータスの読み出しに用いられます．またDMAデータ転送時には，メモリやI/Oから読み出されたデータを一時Z80 DMA内にラッチするのに用いられ，サーチの場合にはこれらのライン上のデータと，マッチ・バイトとが比較されます．

・IEI（インタラプト・イネーブル・イン――入力，“H”で有効）

この信号はIEOとともに用いて，Z80システムの割り込みデイジー・チェインを形成します．この入力信号が“H”であるということは，当該Z80 DMAより優先度の高いペリフェラルが，割り込み要求を出していないことを示します．当該Z80 DMAの割り込み優先順位が最高位にある場合には，このIEI入力を常に“H”にします．

・IEO（インタラプト・イネーブル・アウト――出力，“H”で有効）

この信号はIEI入力が“H”であり，かつ当該Z80 DMAが割り込み要求を持っていないときにのみ“H”となります．換言するとこの信号はIEI入力が“L”であると“L”となり，IEI入力が“H”であっても当該Z80 DMAが割り込み要求を持っていれば“L”となります．

Z80システムの割り込みに関しては，別項で述べられていますのでここでは割愛し，Z80 DMAを含む割り込みデイジー・チェインの例（図24）を示すのみとします．

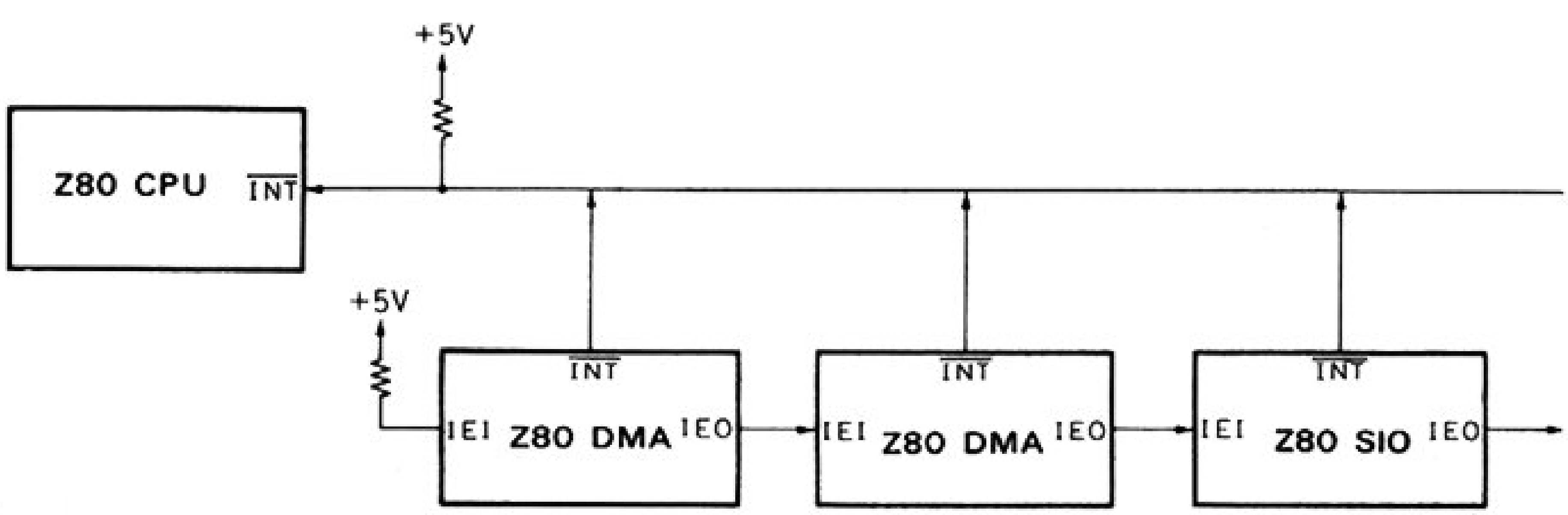
・$\overline{INT}/\overline{PULSE}$（割り込み要求／パルス出力，“L”で有効，オープン・ドレイン型出力）

このピンは2通りの目的，すなわち割り込み要求およびパルス発生用に使われています．割り込み要求線としては，他のZ80ペリフェラルと同様に割り込み要求があり，かつIEIが“H”のときに，この信号は“L”になります（この場合，$\overline{BUSACK}$=“H”）．

パルス発生用としては，256バイトのDMAデータ転送が行われるごとに，$\overline{INT}$ピン上にパルス（“L”）を出力します．この信号は，必ずCPUの$\overline{BUSACK}$が“L”であるときに出力されますので，周辺回路はこの信号と$\overline{BUSACK}$とをANDゲートを通すことにより，パルス出力だけを抽出することができます．

・$\overline{IORQ}$（I/Oリクエスト，“L”で有効，双方向性）

〈図24〉
割り込みデイジー・チェイン



この信号は双方向性であり，DMA データ転送時($\overline{\mathrm{BUSACK}}$="L")には出力となり，それ以外のときは入力となります．入力の場合には，CPU が OUT 命令，または IN 命令を実行した場合に，この信号は"L"になります．

また，Z80 CPU が割り込み認識シーケンスを実行したときにも，この$\overline{\mathrm{IORQ}}$は"L"となり，この場合には$\overline{\mathrm{M1}}$も同時に"L"になります．

他の Z80 DMA が，DMA データ転送をしているときにも，この信号は"L"になりますが，通常はこの状態は当該 DMA にとって何の意味もありません．

当該 Z80 DMA が，DMA データ転送を行うときには，この信号は出力となり，アドレス・バスおよび$\overline{\mathrm{RD}}$または$\overline{\mathrm{WR}}$信号を使用することにより，I/O のデータの読み書きを行います．

・$\overline{\mathrm{M1}}$（マシン・サイクル 1，入力，"L"で有効）

このピンには，Z80 CPU の$\overline{\mathrm{M1}}$信号が直接的に接続されます．$\overline{\mathrm{M1}}$信号は，Z80 CPU が命令のマシン・サイクル 1 （OP コード・フェッチ・サイクル）を実行していることを示すほかに，割り込み認識シーケンスを示すこともあります．Z80 DMA はこの$\overline{\mathrm{M1}}$信号を，RETI 命令の識別，および割り込み認識シーケンスの識別に用いています．

・$\overline{\mathrm{MREQ}}$（メモリ・リクエスト，出力，"L"で有効，3 ステート）

Z80 DMA がメモリを対象とする DMA データ転送を行うときに，この信号の出力バッファはイネーブルされ，メモリのデータを読み書きするのに用いられます．

・$\overline{\mathrm{RD}}$（リード，"L"で有効，双方向性）

Z80 DMA が DMA データ転送をしていないときには，この信号は入力となり，CPU が当該 DMA のリード・レジスタを読むときに用いられます．DMA データ転送を行うときには，この信号は出力となり，メモリまたは I/O からデータを読むのに用いられます．

・$\overline{\mathrm{WR}}$（ライト，"L"で有効，双方向性）

Z80 DMA が DMA データ転送をしていないときには，この信号は入力となり，CPU が当該 DMA に対してデータをライト（プログラミング）するのに用いられます．DMA データ転送時には，この信号は出力となり，メモリまたは I/O に対してデータを書き込むのに用いられます．

・RDY（レディ，入力，有効極性はプログラマブル）

この信号の有効極性（"H"/"L"）はプログラマブルであり，Z80 DMA のライト・レジスタ 5 のビット 3 により指定できます．この信号が有効極性になると，Z80 DMA は，指定されたモード（バイト，バースト，コンティニュアス）での DMA データ転送を行います．

Z80 DMA の コントロール・レジスタ

Z80 DMA を動作させるには，Z80 DMA のライト・レジスタに対して必要な値を書き込む必要があります．また，Z80 DMA の状態を知りたいときには，リード・レジスタの内容を読み込みます．

前者の場合には，OTIR 命令，または OUT 命令が用いられ，後者の場合には INIR 命令，または IN 命令が使用されますが，どちらの場合においても，I/O アドレス・デコーダの Z80 DMA に対する出力が"L"となり，これが$\overline{\mathrm{CE}}/\overline{\mathrm{WAIT}}$ピンに接続されていなければなりません．

以下に Z80 DMA のライト・レジスタ，およびリード・レジスタの中身およびそれらに対するアクセス方法について説明をします．

Z80 DMA のライト・レジスタ (Write Registers)

Z80 DMA は WR0～WR6 という名を持つ，ライト・レジスタが 7 個あります．これらのレジスタは，ベース・レジスタ，およびサブ・レジスタという構成をとっており，ベース・レジスタ中のポインタ・ビットにより，ベース・レジスタに続く，コントロール・ワードを指定するような仕組みになっています．

図25にベース・レジスタとポインタ・ビット，ならびにサブ・レジスタの関係を示します．まず，ベース・レジスタにデータをライトします．このデータ中にはポインタ・ビットというビットがあり，これらのビットが立っていると("1")，その次にライトされるデータは，ポインタ・ビットが示すサブ・レジスタに書き込まれます．

ベース・レジスタ中のポインタ・ビットは，複数ある場合には，右から左へ (LSB → MSB) という順序で後続のサブ・レジスタを指定します．

図25の例では，ポインタ・ビットはビット 3 ～ 6 ですが，この中で"1"が立っているのはビット 3，4，6 です．したがってこのベース・レジスタに続くデータは，サブ・レジスタ 0，1，3 に対するデータであることになります．

Z80 DMA は I/O アドレスを一つしか持ちません($\overline{CE}$)が，じつのベース・レジスタをどのようにして識別するのかというと，データ中のあるビットの組み合わせにより行います．図26にベース・レジスタのアドレッシングを示します．この図からわかるように，Z80 DMA はベース・レジスタ・バイトのビット7，2，1，0の組み合わせにより，WR0～WR7のうちのどれであるのかの識別を行います．

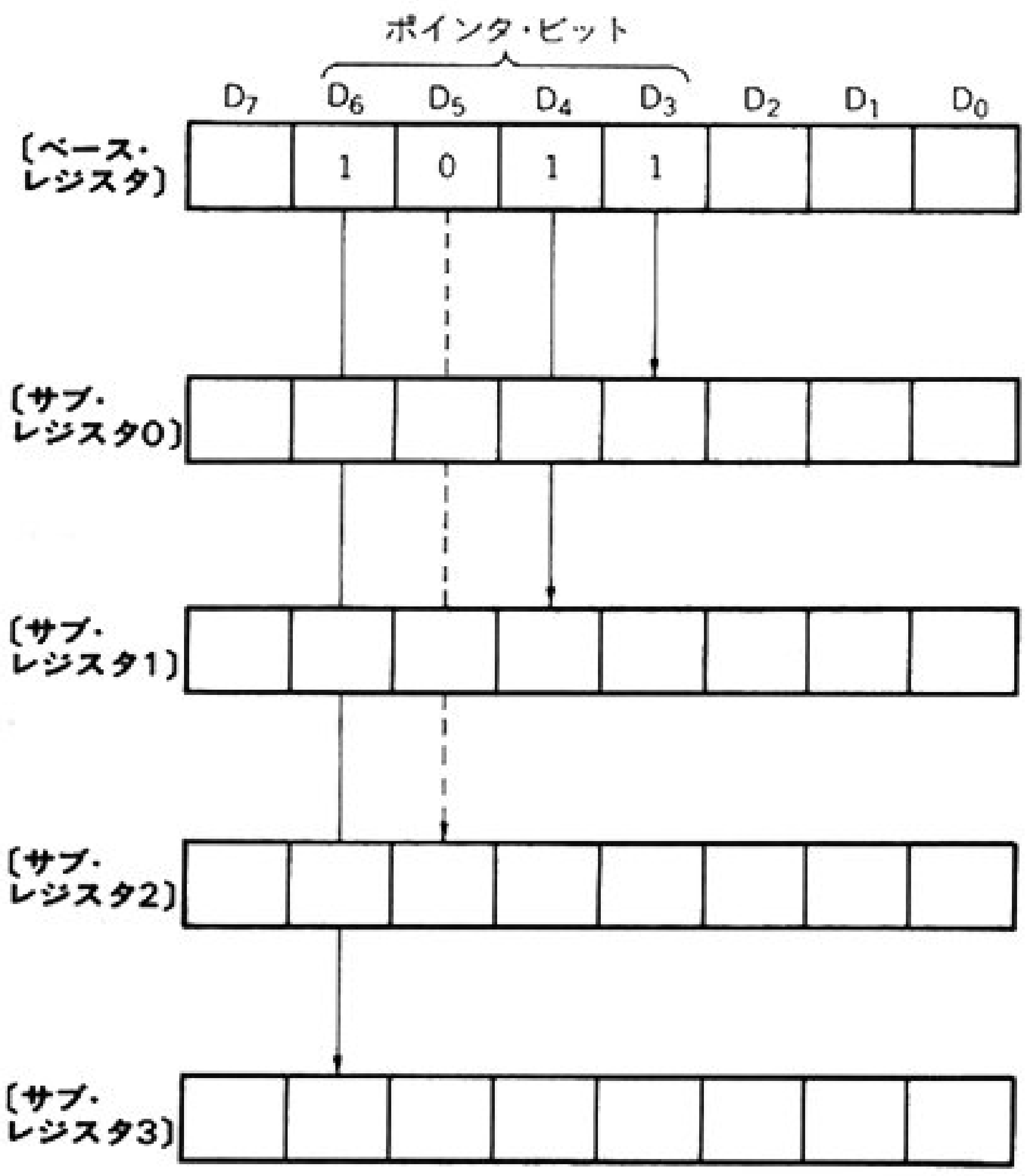
ポインタ・ビットを持っているのはベース・レジスタだけではなく，WR4 中の割り込み制御バイトのように，サブ・レジスタがポインタ・ビットを持っていることもあります．この場合のポインタ・ビットの取り扱いも，ベース・レジスタの場合と同じです．

また WR6 のコマンド BB_{16}(16進，リード・マスク・フォローズ)は，ポインタ・ビットを持っているわけではありませんが，このコマンドの次に続くデータは，リード・マスクでなければなりません．

*

以上でライト・レジスタについての一般的な話を終わり，以降，個々のライト・レジスタの内容について説明をします．

〈図25〉 ポインタ・ビット．ビット5が"0"であるので，サブ・レジスタ2はベース・レジスタ・バイトの後に続かない



〈図26〉 Z80 DMA ベース・レジスタの選択

D_7	D_2	D_1	D_0	セレクトされるベース・レジスタ
0	×	0 1 1	1 0 1	WR0
0	1	0	0	WR1
0	0	0	0	WR2
1	×	0	0	WR3
1	×	0	1	WR4
1	0	1	0	WR5
1	×	1	1	WR6

ライト・レジスタ0(WR0)

図27にライト・レジスタ0の構成を示します．このWR0のベース・レジスタは，ビット7が0でビット0および1がともに0でないという条件により識別されます．また，このWR0は四つのポインタ・ビットを持っており，それぞれに関連した四つのサブ・レジスタを持っています．

■ビット0，1(データ転送形式の指定)

ベース・レジスタ・バイトのビット0，および1により三つのデータ転送形式(トランスファ，サーチ，サーチ／トランスファ)のうちの一つを選択します．

■ビット2(転送データの方向指定)

ベース・レジスタ・バイトのビット2により，転送データの方向を指定します．このビットが"0"であると，ポートB → ポートAの向きにデータが転送され，このビットが"1"であると，ポートA → ポートBの向きにデータが転送されます．

Z80 DMA のポートA，およびポートBはまったく対称な機能を持っています．したがって，たとえばメモリから I/O へ DMA データ転送をしたいような場合，どちらをメモリに指定してもよく，またどちらを I/O に指定してもかまいません．ですからこのビットによる，データ転送方向指定もプログラマの意のままに決めることができます．

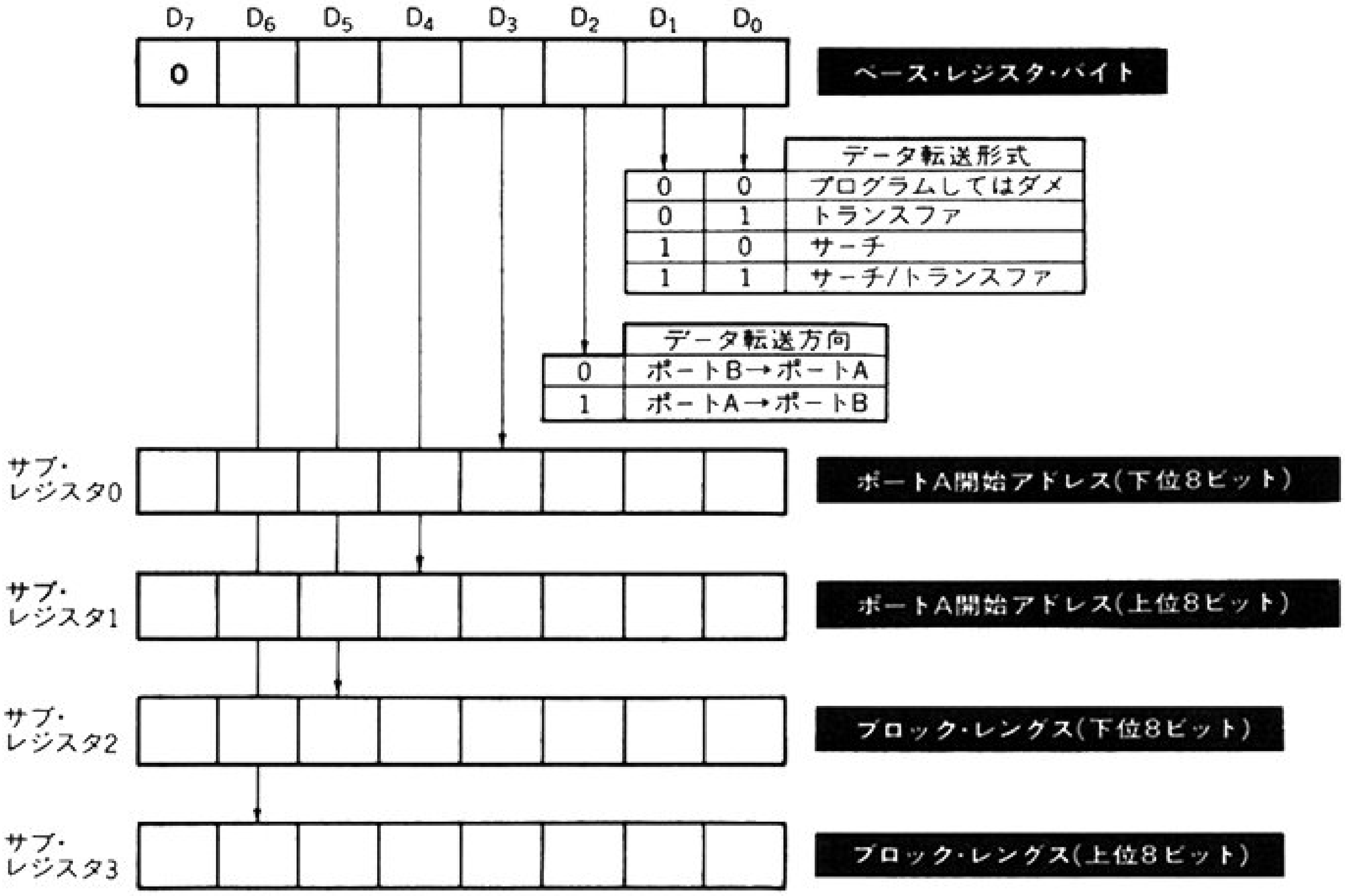
■ビット3～6(ポインタ・ビット)

ベース・レジスタ・バイトのビット3～6はポインタ・ビットであり，後続の四つのサブ・レジスタを指定します．

■サブ・レジスタ0，1(ポートA開始アドレス)

ベース・レジスタ・バイトのビット3，およびビット4によりこれらのレジスタに対するデータが後続することが指定されます．

サブ・レジスタ0はポートA開始アドレスの下位8



〈図27〉
ライト・レジスタ0（WR0）の構成

ビットであり，サブ・レジスタ1は上位8ビットです．ポートAをメモリであると指定する場合には，両サブ・レジスタに対してデータをライトする必要がありますが，ポートAをI/Oと指定する場合には，通常は開始アドレスの下位8ビットをライトするだけで十分です．

もちろん，16ビット・アドレスをI/Oに対しても用いているようなシステムでは，両レジスタにデータをライトする必要があります．

■サブ・レジスタ3，4（ブロック長）

ベース・レジスタ・バイトのビット5，およびビット6により，これらのレジスタに対するデータが後続することを指定します．これらのレジスタは，DMAデータ転送すべきブロックのバイト数を指定するのに用いられます．サブ・レジスタ3はブロック長の下位8ビット，サブレジスタ4は上位8ビットを保持します．

このレジスタには，転送すべきデータのバイト数マイナス1の値をセットします．このレジスタにゼロをセットした場合には $2^{16}+1$(65536) バイトのデータ転送が行われます．したがって，DMAデータ転送を行うブロック長の最小値は，このレジスタに値0001をセットすることにより得られ，ブロック長は2バイトとなります．

なお，パワーON時にはこのレジスタの値は不定なので，たとえ256バイト以下のデータ転送をする場合でも，両レジスタに対してデータをライトする必要があります．また，このレジスタの値はポートAおよびポートBの双方を制御しますので，開始アドレスのようにポートA用，ポートB用といった具合に，2組ある必要はありません．

ライト・レジスタ1（WR1）

図28にライト・レジスタ1の構成を示します．このWR1はベース・レジスタ・バイトのビット7，1，0が"0"で，かつビット2が"1"であることにより識別されます．

■ビット3（ポートAの指定）

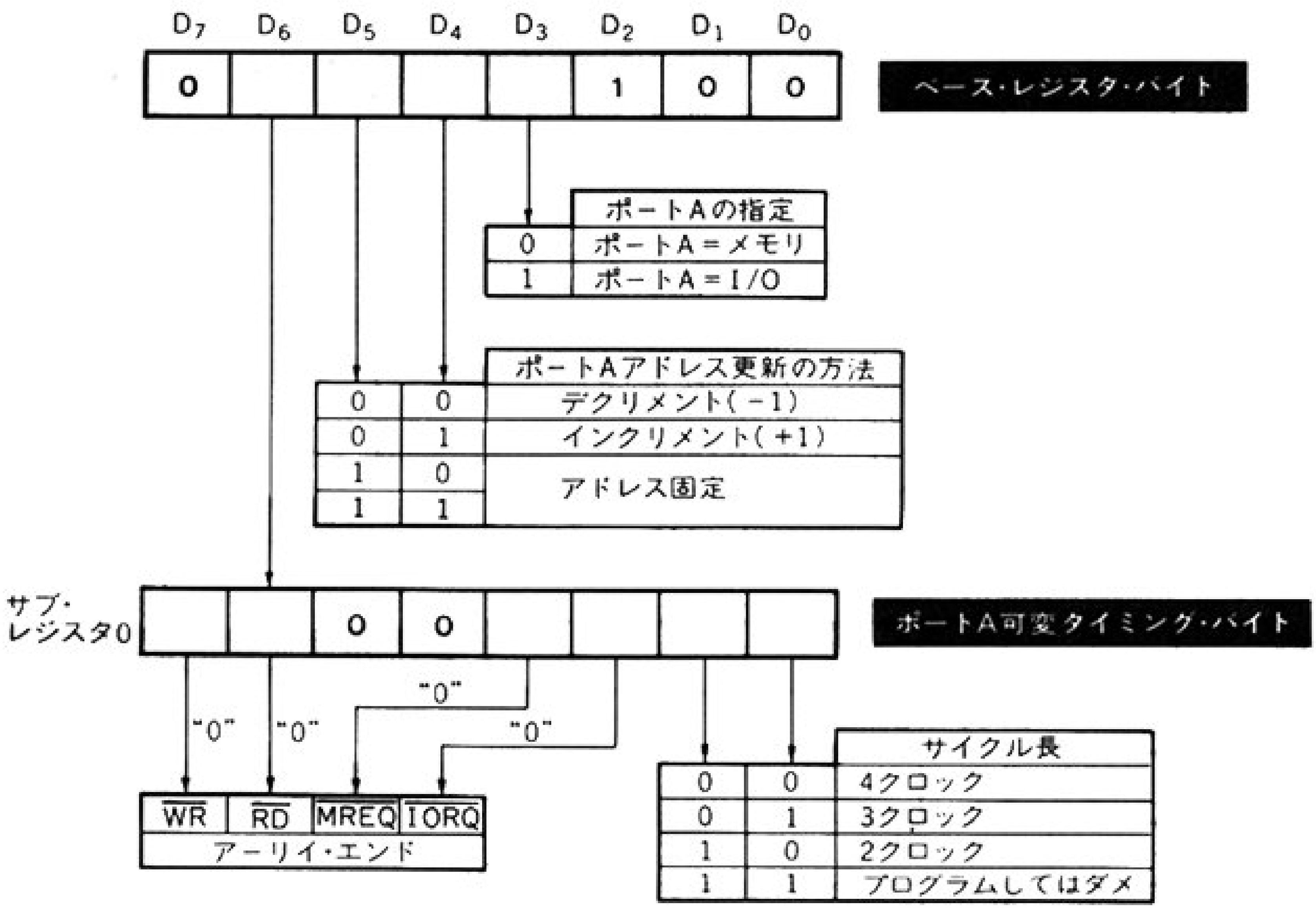
このビットが"0"であると，ポートAはメモリを相手にしてデータ転送をし，"1"であるとI/Oを相手にしてデータ転送をします．メモリと指定すると，DMAデータ転送時にZ80 DMAはポートAに関連したデータ転送のときに $\overline{MREQ}$ を出力し，I/Oと指定すると，$\overline{IORQ}$ を出力します．

■ビット4，5（アドレス更新方法の指定）

ビット4と5により，1バイトのDMAデータ転送が終了するごとに，どのようにポートAアドレスを変化させるかを指定します．ビット4，5が"00"であると，アドレスは1だけ減じられ，ビット4，5が"10"であると1だけ進められます．またビット5が"1"であると，アドレスは固定されたままで変化しません．

通常のメモリ → I/OのDMAデータ転送では，バッファは下位アドレスから上位アドレスの向きにアクセスされますので，ポートAをメモリと指定した場合には，ビット4，5を"10"とします．また，I/Oアドレスは，通常ある一定の値を取り変化しませんので，ポートAをI/Oと指定した場合には，ビット5＝"1"

〈図28〉
ライト・レジスタ1(WR1)の構成



とします．

ごくまれにではありますが，メモリ上のバッファを後ろのほうから前のほうへと使用する場合（バックワード・トランスファ）があります．このときには，ポートAがメモリである場合，ビット4，5＝"00"と指定します．

■ビット6（ポインタ・ビット）

このビットが"1"であると，ベース・レジスタ・バイトの次に，サブ・レジスタ0（ポートA可変タイミング・バイト）に対するデータが続くことを示します．

■サブ・レジスタ0(ポートA可変タイミング・バイト)

ベース・レジスタ・バイトのビット6が"1"であると，ベース・レジスタ・バイトの次にサブ・レジスタ0に対するデータが続くことを示します．このサブ・レジスタ0はポートA可変タイミング・バイトと呼ばれ，ポートAに関連したDMAデータ転送時の制御信号のタイミング，および1データ転送サイクルの期間を指定します．

・ビット0，1(サイクルの長さ)：これら2ビットにより，ポートAに関連したDMAデータ転送サイクル（メモリ・リード／ライト，I/Oリード／ライト）の期間を指定します．この可変タイミング・バイトを使用しないときには，Z80 DMAはZ80 CPUと同じタイミングで，データ転送を行いますので，メモリ・リード／ライト・サイクルは3クロック期間で行い，I/Oリード／ライト・サイクルは4クロック期間で行います．

ところが可変タイミング・バイトを用いることにより，メモリまたはI/Oに対するリード／ライト・サイクルの期間を2～4クロックの範囲で変更することができます．このクロック数の指定を，ビット0およびビット1により行います．

・ビット2，3，6，7（アーリイ・エンド）：可変タイミング・バイトにより，サイクルの長さを変更した場合，これらのビットをセットしないときには，制御信号はクロックの立ち上がり時点にL → Hの変化をしますが，これらのビットをセットすることにより，制御信号がL → Hの変化をするタイミングを半クロック分だけ前にすることができます(アーリイ・エンド)．

ここでいう制御信号とは，$\overline{IORQ}$，$\overline{MREQ}$，$\overline{RD}$，ならびに$\overline{WR}$のことであり，それぞれ可変タイミング・バイトのビット2，3，6，7をセットすることにより，アーリイ・エンドさせることができます．

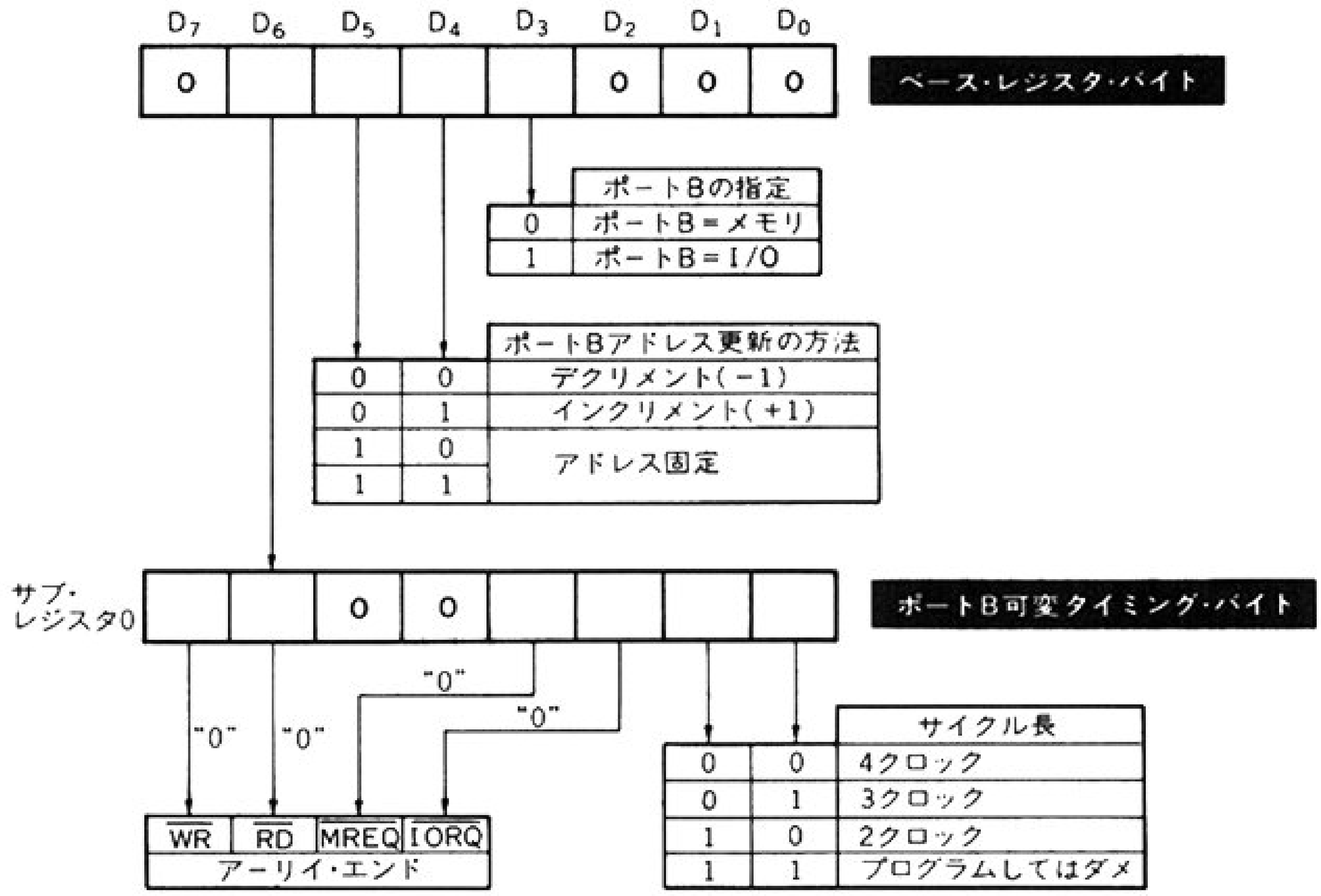
ライト・レジスタ2(WR2)

図29にライト・レジスタ2の構成を示します．このレジスタは，ベース・レジスタ・バイトのビット7，1，0がともに"0"であることにより識別されます．

ライト・レジスタ2の機能はライト・レジスタ1のそれと完全に対称であり，ライト・レジスタ1がポートAの属性と可変タイミングの指定に用いられていたのに対して，ライト・レジスタ2は，ポートBの属性と可変タイミングの指定に用いられます．

したがってライト・レジスタ1の説明中の，ポートAという用語をポートBに置き換えることにより，ライト・レジスタ1の説明がそのまま適用できます．

<図29>
ライト・レジスタ 2 (WR2)の構成



ライト・レジスタ 3 (WR3)

図30にライト・レジスタ3の構成を示します. このレジスタは, ベース・レジスタ・バイトのビット7が"1"で, ビット1, 0がともに"0"であることにより識別されます.

■ビット2(ストップ・オン・マッチ)

このビットはデータ転送形式がサーチ, またはトランスファ／サーチのときに用いられます. このビットが, "1"であるとDMA転送したデータがマッチ・バイト(Match Byte)と一致すると, DMAデータ転送をストップします.

■ビット3 (ポインタ・ビット)

このビットが"1"であると, ベース・レジスタ・バイトの後に, マスク・バイトが続くことを示します.

■ビット4 (ポインタ・ビット)

このビットが"1"であると, ベース・レジスタ・バイトの後に, マッチ・バイトが続くことを示します.

■ビット5 (割り込み・イネーブル)

このビットが"1"であると, Z80 DMAの割り込みが可となります. このビットと同一の機能がWR6によってもサポートされています.

■ビット6 (イネーブルDMA)

このビットが"1"であると, Z80 DMAのDMA動作がイネーブルされ, CPUに対してバス・リクエストを開始します. このビットと同じ機能がWR6によってもサポートされています.

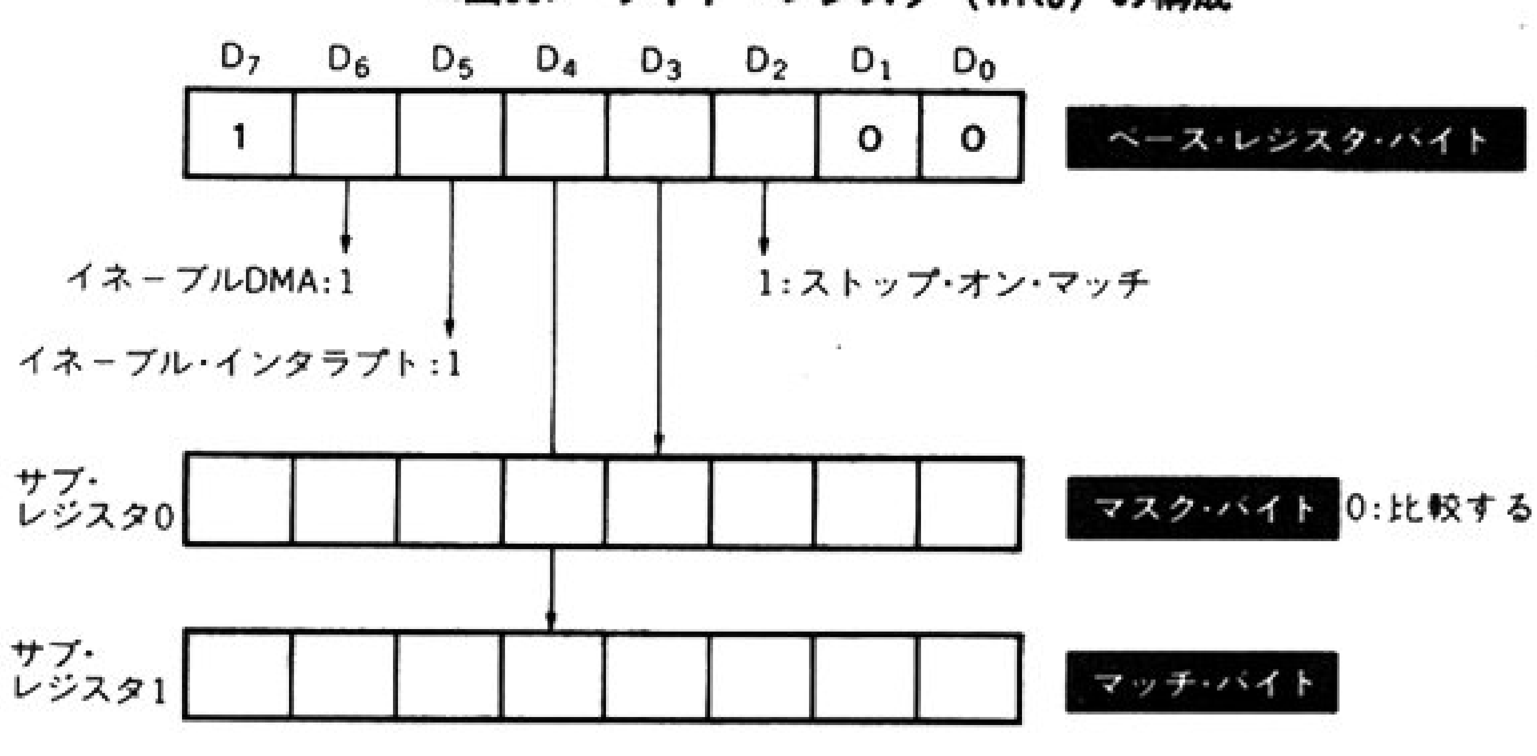
■サブ・レジスタ0 (マスク・バイト)

ベース・レジスタのビット3を"1"にすることにより, サブ・レジスタ0をアクセスすることができ, サーチ動作に関するマスク・バイトを送ることができます.

このマスク・バイトはサーチ中に比較すべきデータ(マッチ・バイト)に対してマスクをかけ, 比較したいビットだけを抽出したり, 比較したくないビットを排除するのに用います.

マスク・バイトのビットが"0"であると比較の対象となり, "1"であるとマスクされ, 比較の対象から

<図30> ライト・レジスタ (WR3) の構成



はずされます．マスクを使用する必要がなく，全ビットを比較の対象とするときには，00_{16}のマスク・バイトを送ります．図31にマスク・バイトの働きを示します．

■サブ・レジスタ1（マッチ・バイト）

ベース・レジスタ・バイトのビット4を“1”とすることにより，サブ・レジスタ1をアクセスすることができ，マッチ・バイトを送ることができます．

このマッチ・バイトは，DMAデータ転送形式がサーチ，またはトランスファ／サーチのときに有効となり，比較すべきデータとなります．このマッチ・バイトは前述のマスク・バイトにより，マスクをかけることができます．

ライト・レジスタ4（WR4）

ライト・レジスタ4は，ベース・レジスタ・バイトのビット7および0がともに“1”で，ビット1が“0”であることにより識別されます．図32にライト・レジスタ4の構成を示します．

■ビット2～4（ポインタ・ビット）

これらのビットが“1”であると，ベース・レジスタのうしろに，サブ・レジスタ0～2がそれぞれ続くことを示します．

■ビット5，6（DMAデータ転送モード）

ベース・レジスタのビット5および6により，DMAデータ転送モード（バイト，コンティニュアス，バースト）の選択を行います．ビット5および6が“00”であると，バイト・モードとなり，“10”であると，コンティニュアス・モードとなり，“01”であると，バースト・モードとなります．

■サブ・レジスタ0，1（ポートB開始アドレス）

ベース・レジスタ・バイトのビット3および4が“1”であると，ベース・レジスタ・バイトの後に，これらのサブ・レジスタに対する値が続くことを示します．

サブ・レジスタ0はポートB開始アドレスの下半バイトを指定し，サブ・レジスタ1はポートB開始アドレスの上半バイトを指定します．

なお，WR0のサブ・レジスタ0，1と同様に，ポートBがI/Oである場合など，ポートBの開始アドレスが下位8ビットのみでよい場合には，サブ・レジスタ1へのロードは不要です．

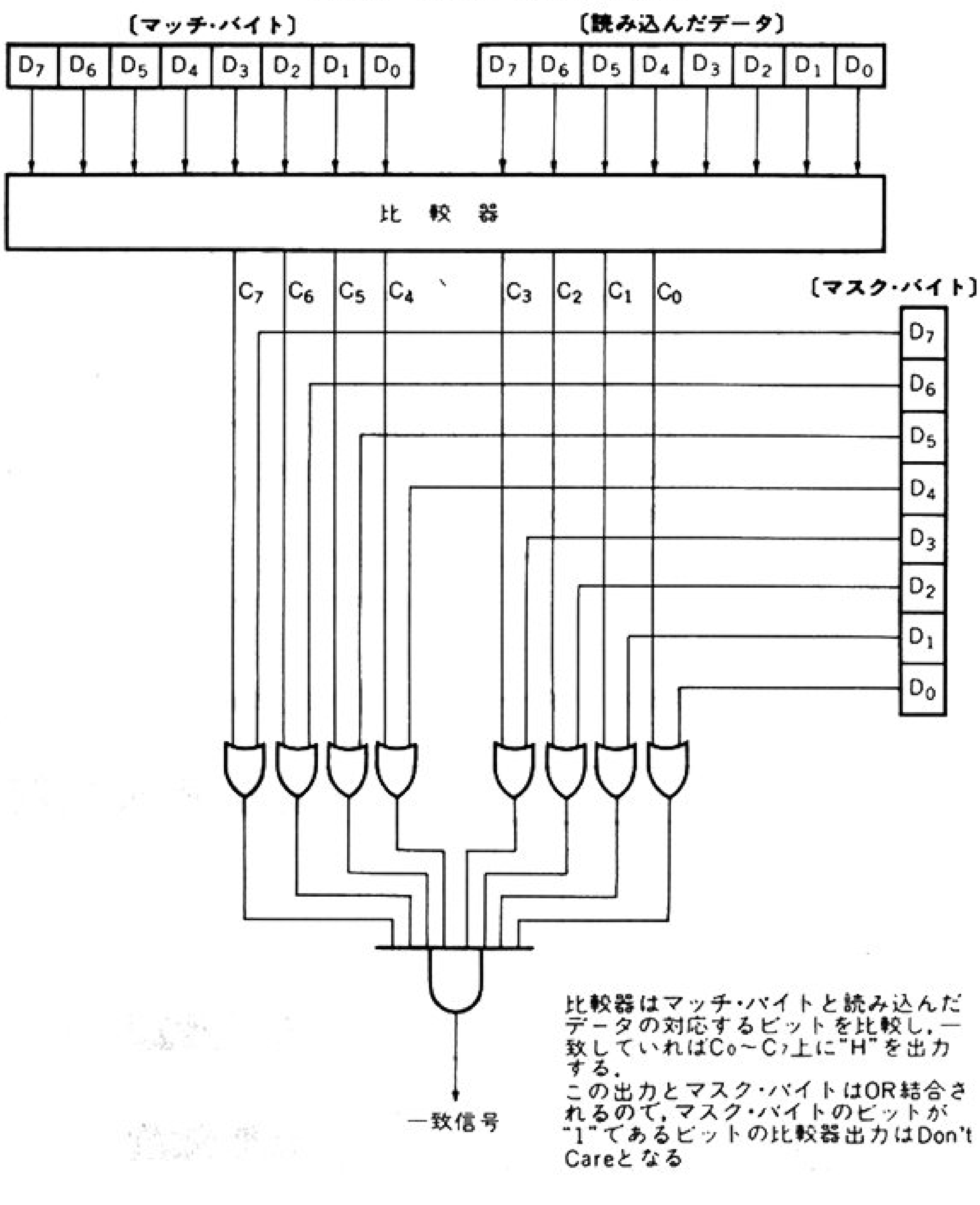
■サブ・レジスタ2（割り込み制御バイト）

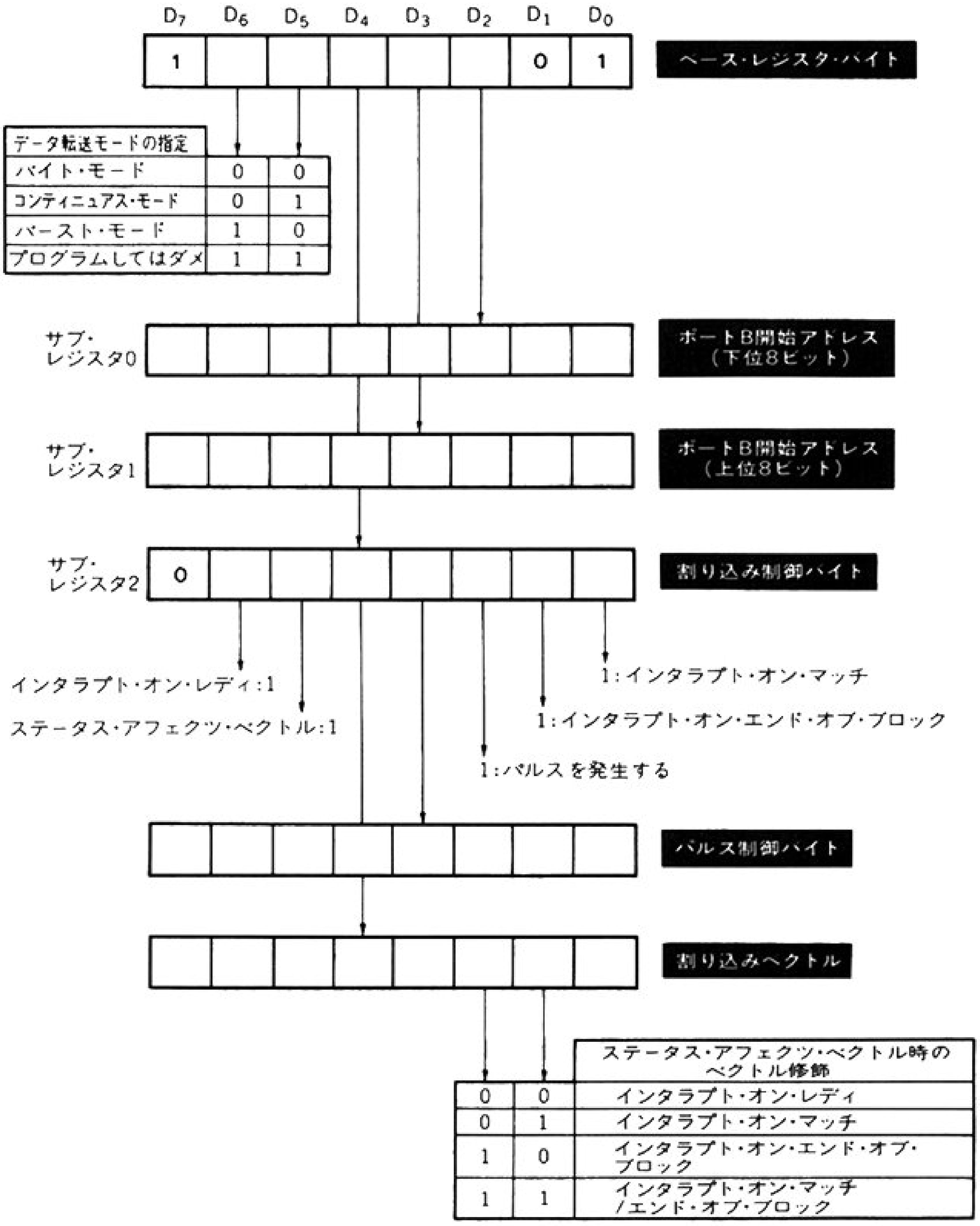
サブ・レジスタ2は割り込み制御バイトという名を持っており，Z80 DMAの割り込みを制御しますが，このほかにパルス制御バイトをサブ・レジスタとして持っており，パルス発生の制御をも行います．

・ビット0（インタラプト・オン・マッチ；マッチ・バイトとの一致検出時に割り込みを発生する）：このビットが“1”であると，サーチまたはトランスファ／サーチの際，DMAデータが転送したデータが，マッチ・バイトと一致した場合に，Z80 DMAは割り込みを発生します．

・ビット1（インタラプト・アット・エンド・オブ・ブロック；エン

〈図31〉 マスク・バイトの働き





〈図32〉
ライト・レジスタ4（WR4）の構成

ド・オブ・ブロック時に割り込みを発生する）：このビットが"1"であると，DMAデータ転送中，バイト・カウンタの値が0になったときに，Z80 DMAは割り込みを発生します．

・**ビット2（パルス発生）**：このビットが"1"であると，パルス制御バイトにより指定されたバイト数のデータを，DMA転送するごとに，$\overline{INT}$ピン上にパルスを発生します．

・**ビット3，4（ポインタ・ビット）**：ビット3が"1"であると，割り込み制御バイトの後に，パルス制御バイトが続くことを示し，ビット4が"1"であると，割り込みベクトルが続くことを示します．

・**ビット5（ステータス・アフェクツ・ベクトル；ステータスにより，ベクトルの値を変化させる）**：このビットには，ステータス・アフェクツ・ベクトル（Status Affects Vector）という名が付いており，"1"であると割り込み発生要因にしたがって割り込みベクトルの値が変化します．

・**ビット6（インタラプト・オン・レディ；レディ検出時に割り込みを発生する）**：このビットが"1"であると，Z80 DMAはRDY入力が有効極性になったのを検出すると，バス・リクエストを出さずに，割り込み要求を発生します．

そして割り込みルーチンが"RETI命令の後にDMAをイネーブルする"というコマンド（WR6の$B7_{16}$）を出した後にRETI命令を実行すると，バス・リクエストが開始されます．

■パルス制御バイト

割り込み制御バイトのビット3が"1"であると，

割り込み制御バイトの後にパルス制御バイトが続くことを示します．

このパルス制御バイトは，一番最初に発生するパルスに対して，この制御バイトで示されるバイト数だけのオフセットを与えます．Z80 DMA のパルス制御部はこのパルス制御バイトとバイト・カウンタの下位8ビットを比較し，両者が一致すると $\overline{\text{INT}}$ ライン上にパルスを出力します．

■割り込みベクトル

割り込み制御バイトのビット4が“1”であると，割り込み制御バイトの後にこの割り込みベクトルが続くことを示します．

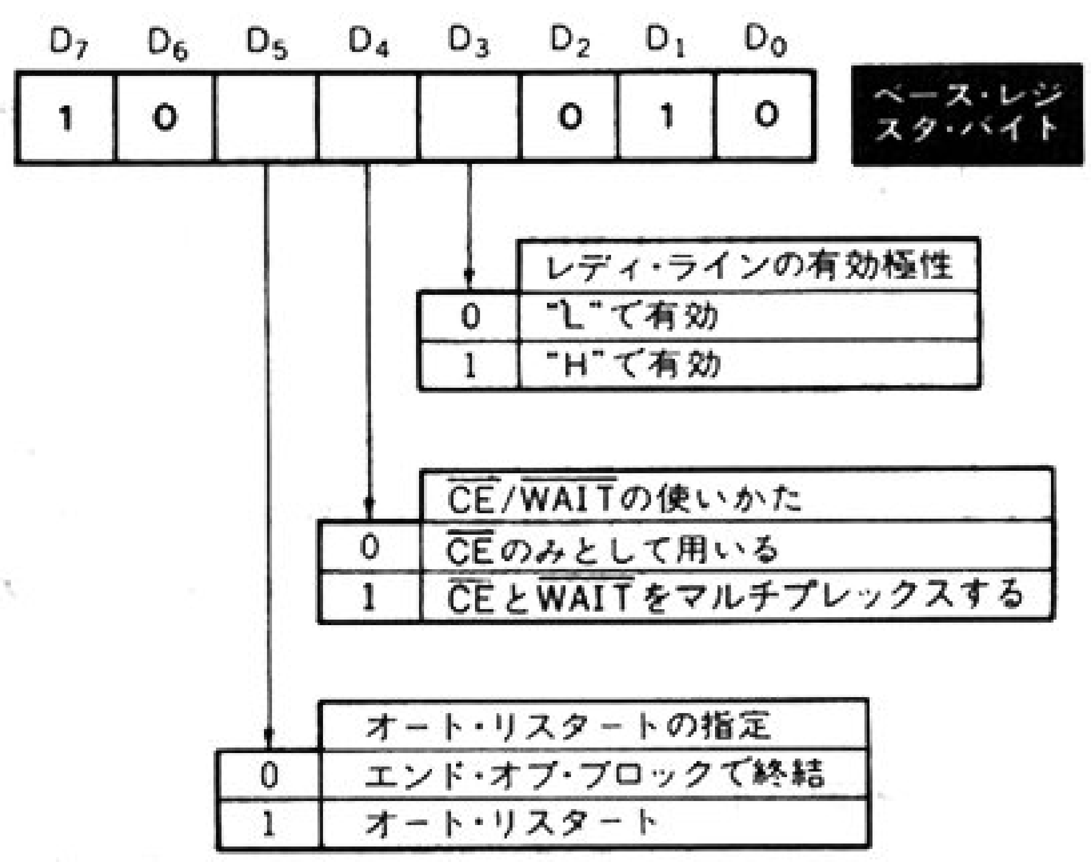
割り込みベクトルは CPU の割り込み認識シーケンス時（$\overline{\text{IORQ}}$＝“L”，$\overline{\text{M1}}$＝“L”）にデータ・バス上に乗せられますが，割り込み制御バイトのビット5が“1”であると，割り込み要因に応じてベクトルのビット1および2が変化します．Z80 CPU は割り込みベクトルから割り込みルーチン・アドレスをアクセスするときに，CPU のインタラプト・レジスタの内容を割り込みルーチン・アドレス・テーブルのアドレスの上位8ビットとして使用するので，同アドレス・テーブルがアドレス $\times 100_{16}$ をまたぐことのないように注意する必要があります．

理由はわかりませんが，ステータス・アフェクツ・ベクトル機能を用い，オート・リスタートでかつ“エンド・オブ・ブロック時の割り込み”と指定したときには，ベクトルは変化しないので注意を要します．

ライト・レジスタ5（WR5）

ライト・レジスタ5はベース・レジスタ・バイトのビット7，および1がともに“1”でビット0が“0”であることにより識別されます．図33にライト・レジスタ5の構成を示します．この図からわかるように，ライト・レジスタ5にはポインタ・ビットはありませんので，サブ・レジスタはありません．

〈図33〉 ライト・レジスタ5（WR5）の構成



■ビット3（RDY の有効極性）

このビットは RDY 入力の有効極性を指定します．このビットが“0”であると，“L”が有効となり，このビットが“1”であると“H”が有効となります．

■ビット4（$\overline{\text{CE}}$/$\overline{\text{WAIT}}$ ピンの使いかた）

このビットは $\overline{\text{CE}}$/$\overline{\text{WAIT}}$ ピンの使いかたを指定します．このビットが“0”であると，$\overline{\text{CE}}$/$\overline{\text{WAIT}}$ ピンは $\overline{\text{CE}}$ 機能のみとなり，このビットが“1”であると，$\overline{\text{CE}}$ および $\overline{\text{WAIT}}$ の両方の機能を果たします．

つまり，$\overline{\text{BUSACK}}$ が“H”のときには $\overline{\text{CE}}$ として働き，$\overline{\text{BUSACK}}$ が“L”のときには，$\overline{\text{WAIT}}$ として働きます．

■ビット5（オート・リスタート）

このビットが“0”であると，エンド・オブ・ブロック（バイト・カウンタ＝ゼロ）を検出すると，DMA データ転送は停止します．このビットが“1”であると，エンド・オブ・ブロック時に，アドレス・レジスタおよびバイト・カウンタ・レジスタの内容が自動的に，アドレス・カウンタ，およびバイト・カウンタにロードされ，DMA 動作が続行されます．

ライト・レジスタ6（WR6）

ライト・レジスタ6はベース・レジスタ・バイトのビット7，1，0がともに“1”であることにより識別されます．

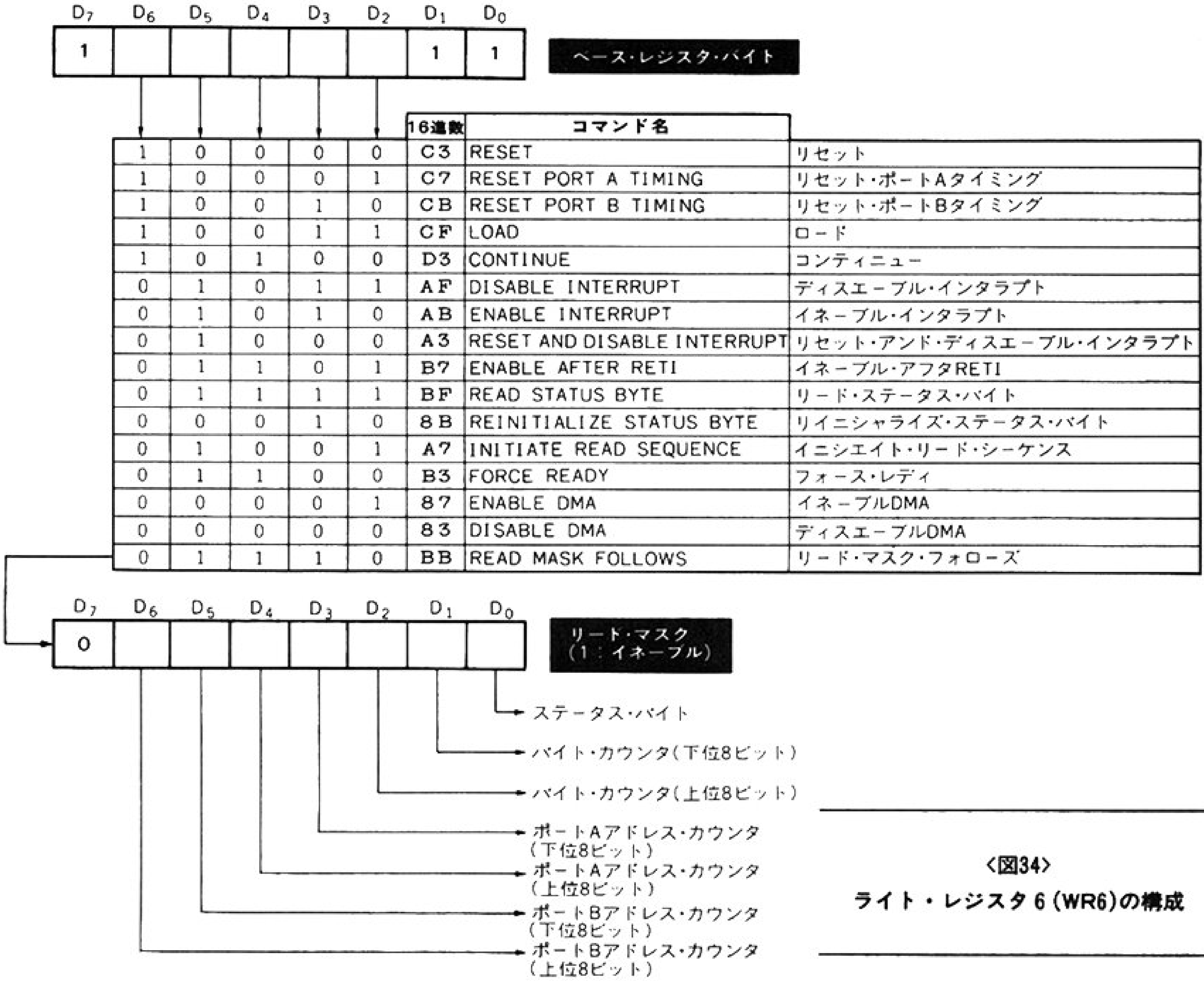
このレジスタにはポインタ・ビットはありませんが，コマンド BB_{16}（リード・マスク・フォローズ）の場合には，ベース・レジスタ・バイトの後にリード・マスクが続きます（図34）．

このレジスタの場合には，ベース・レジスタ・バイトの各ビットが特定の意味を持つようなことはなく，ビット2～6の組み合わせにより機能が分けられます．

■リセット（$C3_{16}$）

このコマンドは，Z80 DMA をリセットするのに用いられます．このコマンドが実行されると，Z80 DMA は次のようなことがらを行います．

❶ 割り込み制御ロジック，およびバス・リクエスト・ロジックをディスエーブルする
❷ 割り込みラッチをリセットする
❸ フォース・レディ状態をクリヤする
❹ オート・リスタート状態をクリヤする



D6	D5	D4	D3	D2	16進数	コマンド名	
1	0	0	0	0	C3	RESET	リセット
1	0	0	0	1	C7	RESET PORT A TIMING	リセット・ポートAタイミング
1	0	0	1	0	CB	RESET PORT B TIMING	リセット・ポートBタイミング
1	0	0	1	1	CF	LOAD	ロード
1	0	1	0	0	D3	CONTINUE	コンティニュー
0	1	0	1	1	AF	DISABLE INTERRUPT	ディスエーブル・インタラプト
0	1	0	1	0	AB	ENABLE INTERRUPT	イネーブル・インタラプト
0	1	0	0	0	A3	RESET AND DISABLE INTERRUPT	リセット・アンド・ディスエーブル・インタラプト
0	1	1	0	1	B7	ENABLE AFTER RETI	イネーブル・アフタRETI
0	1	1	1	1	BF	READ STATUS BYTE	リード・ステータス・バイト
0	0	0	1	0	8B	REINITIALIZE STATUS BYTE	リイニシャライズ・ステータス・バイト
0	1	0	0	1	A7	INITIATE READ SEQUENCE	イニシエイト・リード・シーケンス
0	1	1	0	0	B3	FORCE READY	フォース・レディ
0	0	0	0	1	87	ENABLE DMA	イネーブルDMA
0	0	0	0	0	83	DISABLE DMA	ディスエーブルDMA
0	1	1	1	0	BB	READ MASK FOLLOWS	リード・マスク・フォローズ

〈図34〉
ライト・レジスタ6(WR6)の構成

❺ $\overline{CE}/\overline{WAIT}$機能(WR5のビット4)をリセットし，$\overline{CE}$機能のみとする

❻ ポートA，ポートB両ポートのタイミングを，Z80標準タイミングに戻す

Z80 DMAに対して電源を投入し，プログラミングを行う前には，このコマンドを1回実行する必要があります．また実行中のDMA動作を中断する場合には，このコマンドを6回続けて出す必要があります．

これはWR4が五つのサブ・レジスタを持っていることに起因しており，Z80 DMAがWR4のベース・レジスタ・バイトを受け取ったときに，このコマンドを出しても，サブ・レジスタに対するものと解釈され，WR6コマンドとはとられないことになります．

したがって，どのような状態にあっても，Z80 DMAの動作を中断させたいと思えば，最低6個のリセット・コマンドを出す必要があります．

■リセット・ポートAタイミング($C7_{16}$)

このコマンドはポートAの可変タイミング・バイトをリセットし，Z80標準タイミングに戻します．

■リセット・ポートBタイミング(CB_{16})

このコマンドはポートBの可変タイミング・バイトをリセットし，Z80標準タイミングに戻します．

■ロード(CF_{16})

このコマンドを出すと，アドレス・レジスタの内容がアドレス・カウンタにロードされ，バイト・カウンタがクリヤされます．またこのコマンドにより，フォース・レディ(Force Ready)状態もリセットされます．

このコマンドにより，ロードされるのはソースと指定されたポートのアドレス・カウンタだけであり，デスティネーションと指定されたポートのアドレス・カウンタにロードされるのは，このアドレス・カウンタの値を最初に更新(インクリメント/デクリメント)する時点です．

したがって，デスティネーションと指定されたポートが，“アドレス固定”とプログラミングされているときには，デスティネーション・ポートのアドレス・カウンタにロードする機会がありません．このような状

態はメモリからI/Oへデータ転送をするように，Z80 DMAをプログラミングする際に生じます．

デスティネーション・ポートが“アドレス固定”である場合には，一種のプログラミング・テクニックを使って，デスティネーション・ポートのアドレス・カウンタにアドレス・レジスタの内容をロードします．ここでは説明のため，ポートAをソース（可変アドレス），ポートBをデスティネーション（固定アドレス）とします．

プログラムはまず，ポートBのアドレス・レジスタ（WR4）に対して値をロードします．次に WR0 のビット2を“0”にして，一時的にソース・ポートとデスティネーション・ポートをスワップします．ここでロード・コマンドを出すことにより，一時的にソース・ポートとなったポートBのアドレス・カウンタに対して，アドレス・レジスタの内容がロードされます．

次にポートAのアドレス・レジスタ（WR0）に対して値をロードし，この際 WR0 のビット2を“1”にして，各ポートを本来のソース／デスティネーションに戻します．そして再度ロード・コマンドを用いることにより，ポートAのアドレス・カウンタにアドレス・レジスタの内容をロードします．

以上のようなプログラミング・テクニックを，Z80 DMAでは“アドレス固定デスティネーション・ポート・プログラミング（Fixed-Address Destination Port Programming）”と呼んでいます．

図35にポートAおよびポートBを“アドレス固定”のデスティネーションと設定したときの，プログラミング・シーケンスを示します．

■コンティニュー（Continue――D3₁₆）

このコマンドを出すと，Z80 DMA はバイト・カウンタをクリヤし，エンド・オブ・ブロックやサーチ時のマッチ検出により中断していた DMA 動作を続行します．このコマンドにより，バイト・カウンタはクリヤされますが，アドレス・カウンタの値は変化しません．

このコマンドは複数ブロックのデータを DMA データ転送し，かつ各ブロックの切れ目を知りたいようなときに使います．エンド・オブ・ブロックにより各ブロックの終了時に割り込みを発生させ，割り込みルーチン内でこのコンティニュー・コマンドを実行することにより，バッファには各ブロックのデータが連続的に転送されます．

■ディスエーブル・インタラプト（AF_{16}）

このコマンドは，Z80 CPU 以外の CPU のもとで，Z80 DMA を動作させる場合に，Z80 CPU の割り込み認識シーケンスを模擬するために用います．

Z80 DMA が割り込みをかけたときに，割り込みルーチンの始めにこのコマンドを出すと，$\overline{INT}$ ラインの状態は“H”に戻りますが，さらに次の割り込みを発生するようなことはないようになっています．割り込みルーチンの終わりのほうで，イネーブル・インタラプト・コマンド（AB_{16}）を出すと，次の割り込みの発生が可能となります．

■イネーブル・インタラプト（AB_{16}）

このコマンドは Z80 DMA の割り込み制御ロジックをイネーブルし，割り込み発生を可能にします．ディスエーブル・インタラプト・コマンドを使わない場合には，このコマンドは一度だけ出しておけばよく，割り込みルーチンの終わりに RETI 命令を実行することにより，自動的に次の割り込みが可能となります．

ディスエーブル・インタラプト・コマンドを用いた場合には，もちろんこのコマンドにより割り込みをイ

〈図35〉 固定アドレス（Fixed Address）・プログラミング

順番	コマンド
1	ポートAアドレスをWR0に書き込む
2	ポートAを一時的にソースであると設定する (WR0のビット2=1)
3	ロード・コマンド(CF_{16})により，ポートAアドレスをアドレス・カウンタにロードする
4	ポートB開始アドレスをWR4に書き込む
5	ポートAをデスティネーションに戻す (WR0のビット2=0)
6	ロード・コマンド(CF_{16})により，ポートB(ソース)開始アドレスをアドレス・カウンタにロードする

〔ポートAがアドレス固定デスティネーションのとき〕

順番	コマンド
1	ポートBアドレスをWR4に書き込む
2	ポートBを一時的にソースであると設定する (WR0のビット2=0)
3	ロード・コマンド(CF_{16})により，ポートBアドレスをアドレス・カウンタにロードする
4	ポートA開始アドレスをWR0に書き込む
5	ポートBをデスティネーションに戻す (WR0のビット2=1)
6	ロード・コマンドにより，ポートA(ソース)開始アドレスをアドレス・カウンタにロードする

〔ポートBがアドレス固定デスティネーションのとき〕

ネーブルしないと次の割り込みは発生しません．なお，このコマンドは WR3 のビット5を“1”にしたとまったく同じ機能を果たします．

■リセット・アンド・ディスエーブル・インタラプト($A3_{16}$)

このコマンドは 8080A や 8085CPU のように，割り込み認識信号 ($\overline{INTA}$) を持っているが，RETI 命令を持っていないような CPU と，Z80 DMA をインターフェースするときに用いられます．

割り込みルーチンの終わりにこのコマンドを出し，ついでイネーブル・インタラプト・コマンドを出すことにより，Z80 CPU が RETI 命令を実行したときと同じ効果を Z80 DMA に対して与えることができます．なおこのコマンドを実行すると，フォース・レディ状態は解除されます．

■イネーブル・アフター RETI ($B7_{16}$：RETI 命令実行後に DMA をイネーブルする)

このコマンドは Z80 DMA がインタラプト・オン・レディ (WR4) とプログラムされているときにのみ用いられます．この場合には，Z80 DMA はレディ・ラインの状態が有効極性になったのを検出すると，バス・リクエストを行わず，割り込みを発生します．

割り込みルーチンにてこのコマンドを実行し，最後に RETI 命令を実行したときになってはじめて，Z80 DMA はバス・リクエストを出します．

■リード・ステータス・バイト (BF_{16})

このコマンドを出すと，このコマンドの次に出される Z80 DMA に対するリード命令は，ステータス・バイトに対するアクセスであることを宣言します．

■リイニシャライズ・ステータス・バイト(Reinitialize Status Byte：$8B_{16}$)

このコマンドは Z80 DMA のステータス・バイトのビット4，および5をイニシャライズ (“1”にする) します．このコマンドを出した後のステータス・バイトは図36のようになります．

このコマンドを出す前には，ディスエーブル DMA コマンドなどを出して，Z80 DMA の動作を確実に停止させておく必要があります．そうしないとマッチまたはエンド・オブ・ブロックで停止していた DMA が，このコマンドによりステータスをクリヤされると，再びバス・リクエストを始めて走り出すようなことが起こり得ます．

ステータス・バイトのビット3 (割り込みペンディング) は，割り込み認識シーケンスが実行されるか，リセット・アンド・ディスエーブル・インタラプト・

〈図36〉 リイニシャライズ・ステータス・バイト・コマンド($8B_{16}$)を出した後のステータス・バイトの状態．ビット4，およびビット5がリセット (“1”) される

ビット	値	意　味
0	1/0	1：DMA動作が行われた 0：DMA動作が行われなかった
1	1/0	0：レディ・ラインが有効レベルにある 1：レディ・ラインが無効レベルにある
2	×	Don't Care
3	1/0	1：ペンディング・インタラプトがない 0：ペンディング・インタラプトがある
4	1	マッチが検出されなかった
5	1	エンド・オブ・ブロックが検出されなかった
6	×	Don't Care
7	×	Don't Care

コマンドにより，クリヤされます．ビット0 (DMA オペレーション)はロード・コマンドにより，クリヤされます．

■リード・マスク・フォローズ (BB_{16})

このコマンドはこのコマンドの後に，リード・マスクが続くことを示します．

リード・マスクは7ビットで構成され (ビット7は常にゼロ)，各ビットにより Z80 DMA に対するリード命令により読み取られる，リード・レジスタ (RR0～RR6) を指定します．ビット0～6が RR0～RR6 に対応し，“1”が立っているビットに対応するリード・レジスタが読み取られます．

リード命令により読み取られる順序は，ビット0 (RR0) → ビット6 (RR6) であり，マスクに “1” が立っていないリード・レジスタは読み飛ばされます．図37にリード・マスクと読み取られるレジスタの関係を示します．

■イニシエイト・リード・シーケンス ($A7_{16}$)

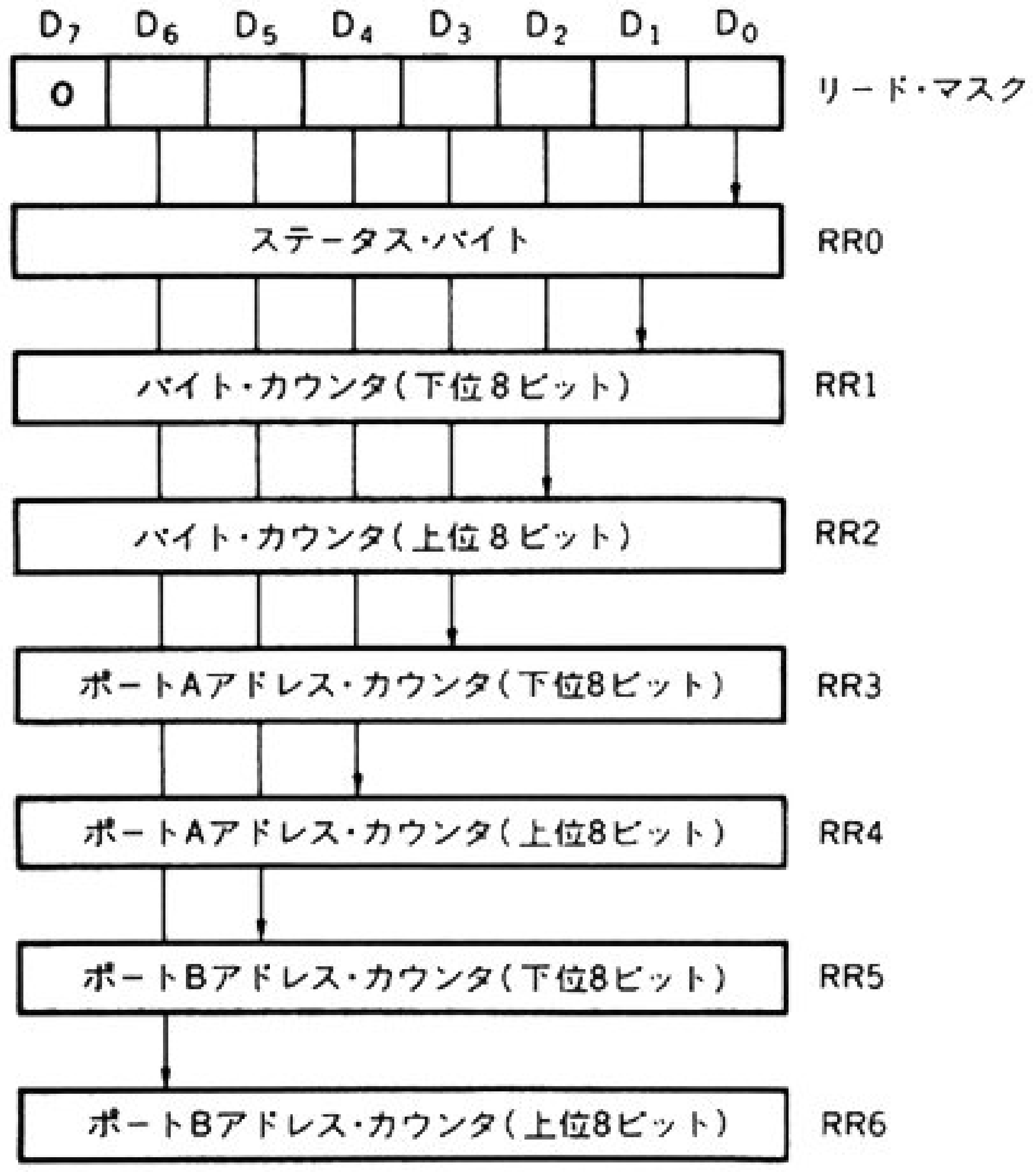
このコマンドを出すと，リード・シーケンスがリセットされ，次に出されたリード命令により読み取られるリード・レジスタは，リード・マスクの LSB 側からみて最初に “1” が立っているビットに対応するリード・レジスタとなります．

ふつうはリード・マスクをロードした直後にこのコマンドを出して，リード・シーケンスをリセットしておきます．

■フォース・レディ (Force Ready：$B3_{16}$)

このコマンドは，Z80 DMA の外部回路 (I/O) がレ

〈図37〉 リード・マスクと読み取られるレジスタの関係．ライト・レジスタのポインタ・ビットと同様に，リード・マスクはビット0 → ビット6の順に処理されていく



ディを出す代わりに，プログラムによりレディ状態を作り出すのに用いられます．メモリ ↔ メモリ，メモリ・サーチなどの動作ではI/Oを必要としませんので，この機能は外部回路を減らすためにも有効です．

このコマンドによりDMA動作を起動すると，WR4で指定したDMA転送モードとは無関係に，コンティニュアス・モードで動きます．またこのフォース・レディを用いるときには，レディ・ラインの有効極性(WR5のビット3)はどちらであってもかまいません．

このフォース・レディ状態は以下のような場合に解除されます．

❶ リセット・コマンドを受けたとき
❷ ロード・コマンドを受けたとき
❸ リセット・アンド・ディスエーブル・インタラプト・コマンドを受けたとき
❹ エンド・オブ・ブロック検出によりストップしたとき
❺ マッチ・バイト検出によりストップしたとき
❻ その他，DMAがバス・リクエストを解除したとき

■イネーブルDMA（87_{16}）

このコマンドはZ80 DMAのバス・リクエスト・ロジックをイネーブルします．またこのコマンドの機能はWR3のビット6と同じです．

Z80 DMAはこのコマンド，およびビット6が"1"であるWR3に対するコマンド以外の他のすべてのコマンド（リードも含めて）が出されると，バス・リクエスト・ロジックをディスエーブルします．したがってZ80 DMAを動作させるときには，常にこのコマンドを最後に出す必要があります．

■ディスエーブルDMA（83_{16}）

このコマンドは何らかの理由で，DMA動作を中断するのに用います．

Z80 DMAのリード・レジスタ (Read Registers)

Z80 DMAにはDMA動作の実行状態または終了状態を知るために7個のリード・レジスタを持っています．これらにはRR0～RR6という名前が付けられています．図38にリード・レジスタの構成を示します．

リード・レジスタに対するアクセスのしかたは二つあります．すなわち，

❶ リード・ステータス・バイト・コマンド（BF_{16}）
❷ イニシエイト・リード・シーケンス・コマンド（$A7_{16}$）

リード・ステータス・バイト・コマンドによる方法は，このコマンドをWR6にライトした後に，Z80 DMAに対してリード命令を出すと，ステータス・バイト(RR0)にアクセスできます．イニシエイト・リード・シーケンス・コマンドを用いると，そのあとのリード命令（複数）により，リード・マスクにより指定した，リード・レジスタを全部読むことができます．

以下に個々のリード・レジスタについて説明をします．

ステータス・バイト（RR0）

■ビット0（DMAオペレーション）

このビットが"1"であると，Z80 DMAがバス・リクエストを行ったことを示します．このビットはロード・コマンドにより，クリヤされます．

■ビット1（レディ・アクティブ）

このビットが"0"であると，レディ入力が有効極性にあることを示します．したがってこのビットはレディ入力の状態をそのまま示しているのではなく，レディ入力の有効極性（WR5のビット3）とレディ入力との排他的論理和（Ex-OR）を反転したもの，ということができます．

■ビット2（Don't Care）

■ビット3（インタラプト・ペンディング）

このビットが"0"であると，Z80 DMAがペンディング・インタラプト(CPUにまだ受け付けられない割り込み)を持っていることを示します．このビットおよび以下のビット4および5は"0"で有効ですので注意を要します．

■ビット4（マッチ検出）

このビットが"0"であると，最後に出されたリセット・コマンドまたはリイニシャライズ・ステータス・バイト・コマンドの後に，マッチが検出されたことを示します．このビットは上記二つのコマンドにより，クリヤ（"1"になる）されます．

■ビット5（エンド・オブ・ブロック検出）

このビットが"0"であると，最後に出されたリセット，ロード，コンティニュー，リイニシャライズ・ステータス・バイト・コマンドの後に，エンド・オブ・ブロック状態が検出されたことを示します．このビットは前記のコマンドにより，クリヤ（"1"になる）されます．

■ビット6，7（Don't Care）

バイト・カウンタ（RR1, RR2）

このカウンタ(16ビット)は，DMAデータ転送が行われたデータのバイト数を示します．このカウンタはロード（CF_{16}），コンティニュー（$D3_{16}$），ならびにリセット（$C3_{16}$）コマンドによりゼロ・クリヤされ，1バイトのDMAデータ転送が行われるごとに，1だけ進められます．

DMAデータ転送中，このカウンタの値はブロック・レングス・レジスタ（WR0）の値と比較され，一致が検出されるとDMAデータ転送はストップします．

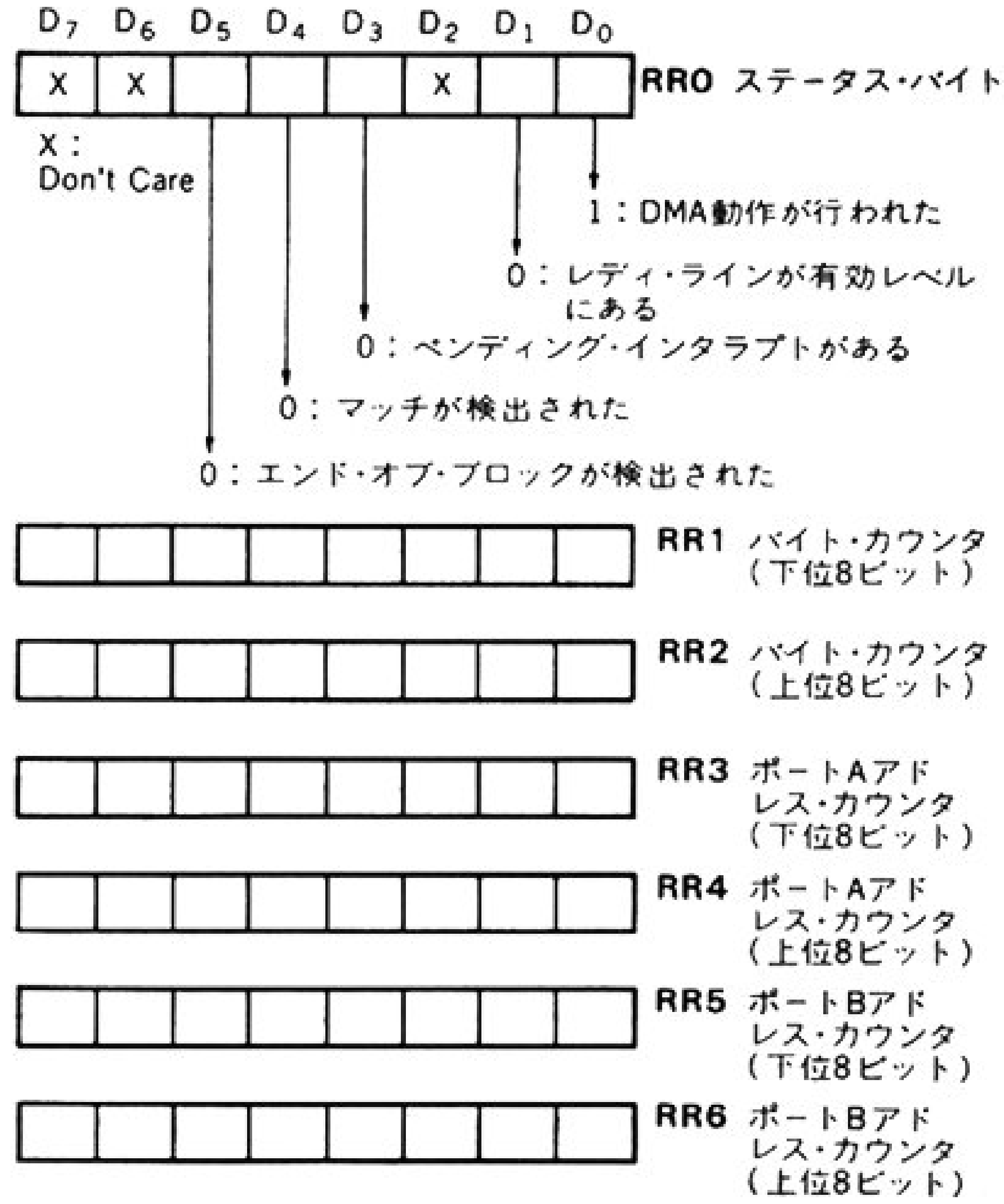
Z80 DMAはデータ転送サイクルを完全に終えてからストップしますので，メモリまたはI/Oからリードしたデータのバイト数と，メモリまたはI/Oに対してライトしたデータのバイト数は常に等しくなります．

このことはサーチ／トランスファ形式のデータ転送に対してもいえ，マッチ検出によりDMAデータ転送が終了する場合においても，ライト動作が終えてからストップします．

ポートAアドレス・カウンタ(RR3, RR4)

このカウンタはDMAデータ転送終了時のポートAアドレスを示します．ロード・コマンド（CF_{16}）により，このカウンタにはポートAアドレス・レジスタ（WR0）の値がロードされ，DMAデータ転送が1バ

〈図38〉 リード・レジスタの構成



イト行われるごとに1だけ更新（インクリメント／デクリメント）されます．

もちろん，アドレス固定（WR1のビット4，5）とプログラミングされているときには変化しません．

ポートBアドレス・カウンタ(RR5, RR6)

このカウンタはDMAデータ転送終了時のポートBアドレスを示します．ロード・コマンド（CF_{16}）により，このカウンタにはポートBアドレス・レジスタ（WR4）の値がロードされ，DMAデータ転送が1バイト行われるごとに，1だけ更新（インクリメント／デクリメント）されます．

もちろん，アドレス固定（WR2のビット4，5）とプログラミングされている場合には変化しません．

DMAデータ転送終了時のアドレス・カウンタ，およびバイト・カウンタの値

前項の説明では，DMAデータ転送終了時のアドレス・カウンタ(RR3～RR6)，およびバイト・カウンタ(RR1, RR2)の値はDMAデータ転送されたバイト数だけ，始めの値（ロード・コマンド実行時）より変化するように述べましたが，実際にはそうではないことがあります．

つまり，バイト・カウンタの値は期待すべき値より1だけ多いことや(エンド・オブ・ブロック時)，1だけ少ないこと（マッチ検出時）があります．またアド

〈図39〉
DMA 動作終結時のカウンタの値（エンド・オブ・ブロック検出時）

データ転送形式	データ転送モード	ブロック・レングス・レジスタの値	転送されるバイト数	バイト・カウンタの値	ソース・ポートのアドレス・カウンタの値	デスティネーション・ポートのアドレス・カウンタの値
トランスファ	バイト	N	N+1	N	As±(N+1)	As±(N)
	バースト	N	N+1	N	As±(N+1)	As±(N)
	コンティニュアス	N	N+1	N	As±(N+1)	As±(N)
サーチ	バイト	N	N+1	N	As±(N+1)	
	バースト	N	N+1	N	As±(N+1)	
			N+2*	N+1*	As±(N+2)*	
	コンティニュアス	N	N+1	N	As±(N+1)	
			N+2*	N+1*	As±(N+2)*	

As：開始アドレス　＊：2サイクル可変タイミングを用い，N+1バイト目のデータを転送した時点にレディ・ラインが有効レベルであるときに起こる

レス・カウンタ，とくにソース・ポートのアドレス・カウンタの値も1または2だけ多く進められていることがあります．

これはデータ転送速度を上げるために，Z80 DMAのDMAデータ転送機構が，高度にパイプライン化され，リード・データに対する先読みを行っているからです．**図39**および**図40**に，それぞれエンド・オブ・ブロック，およびバイト・マッチによりDMAデータ転送が終結したときの，バイト・カウンタおよびアドレス・カウンタの値を示します．

図39からわかるように，エンド・オブ・ブロックにより終結した場合，トランスファのときにはバイト・カウンタの値はブロック・レングス・レジスタの値と一致していますが，ソース・ポートのアドレス・カウンタの値は1だけ多く変化します．

サーチの場合にも通常は，アドレス・カウンタ，およびバイト・カウンタの値はトランスファの場合と同じ値を示しますが，バースト・モードまたはコンティニュアス・モードにおいて，2サイクル・タイミング（可変タイミング）を用いた場合には，バイト・カウンタの値はブロック・レングス・レジスタの値より1だけ多くなり，ソース・ポートのアドレス・カウンタの値は2だけ多くなることがあります．

バイト・マッチにより終結した場合(**図40**)，トランスファ／サーチのときには，ソース・ポートのアドレス・カウンタの値はマッチ・バイトを指していますが，バイト・カウンタの値はマッチ・バイトまでの距離より1だけ少ない値を示しています．

ここで注意すべきことは，デスティネーション・ポートのアドレス・カウンタの値の進みかたはソース・ポートのそれよりも1だけ小さくなっています．したがってマッチ・バイトはデスティネーション・ポートに転送されないということになります．

サーチの場合には，バイト・モードではトランスファ／サーチのときと同じ値を示しますが，バースト・モードまたはコンティニュアス・モードのときには，マッチが検出された直後のレディ・ラインの状態により，アドレス・カウンタ，およびバイト・カウンタの値は異なります．

マッチが検出された直後にも，レディ・ラインの状態が有効レベルにあれば，さらに1バイトのデータ転送が行われるため，両カウンタの値は1だけ多くなります．

Z80 DMAのイニシャライズ

Z80 DMAを動作させるためには，まず一連のデータ（コマンド・チェイン）をZ80 DMAに対して送る必要があります．Z80 DMAにはデフォールト（Default）状態という概念はありませんので，関連するレジスタすべてに対して値をセットする必要があります．

図41にZ80 DMAをどのような動作に使う場合にでも適用できる，イニシャライゼーション・コマンド・チェインを示します．

このコマンド・チェインの長さは36バイトになっていますが，レジスタHLにコマンド・チェインの先頭アドレスをセットし，レジスタBにその長さを，またレジスタCにZ80 DMAのI/Oアドレスをセットすれば，一つのOTIR命令により，イニシャライズを完了することができます．

図41のコマンド・チェインは「フル装備」ですが，自分の応用にとって不要なコマンドは省略することができます．たとえばメモリ・サーチを行うときには，デスティネーション・ポートのアドレス・レジスタに値を書き込む必要はありません．

データ転送形式	データ転送モード	マッチが検出されるバイト番号	転送されるバイト数	バイト・カウンタの値	ソース・ポートのアドレス・カウンタの値	デスティネーション・ポートのアドレス・カウンタの値
トランスファ/サーチ	バイト	M	M	M−1	As ± (M)	As ± (M−1)
	バースト	M	M	M−1	As ± (M)	As ± (M−1)
	コンティニュアス	M	M	M−1	As ± (M)	As ± (M−1)
サーチ	バイト	M	M	M−1	As ± (M)	
	バースト	M	M+1	M	As ± (M+1)	
			M*	M−1*	As ± (M)*	
	コンティニュアス	M	M+1	M	As ± (M+1)	
			M*	M−1*	As ± (M)*	

As：開始アドレス　　＊：マッチが検出された時点にレディ・ラインが有効レベルになかった場合

〈図40〉 DMA 動作終結時のカウンタの値(マッチ検出時)

またサーチを行わない応用ではマッチ・バイト，およびマスクは不要ですし，可変タイミングを用いないときには，可変タイミング・バイトをロードする必要はありません．さらにフォース・レディ・コマンドは外部からレディ信号が供給される場合にはあってはなりません．

＊

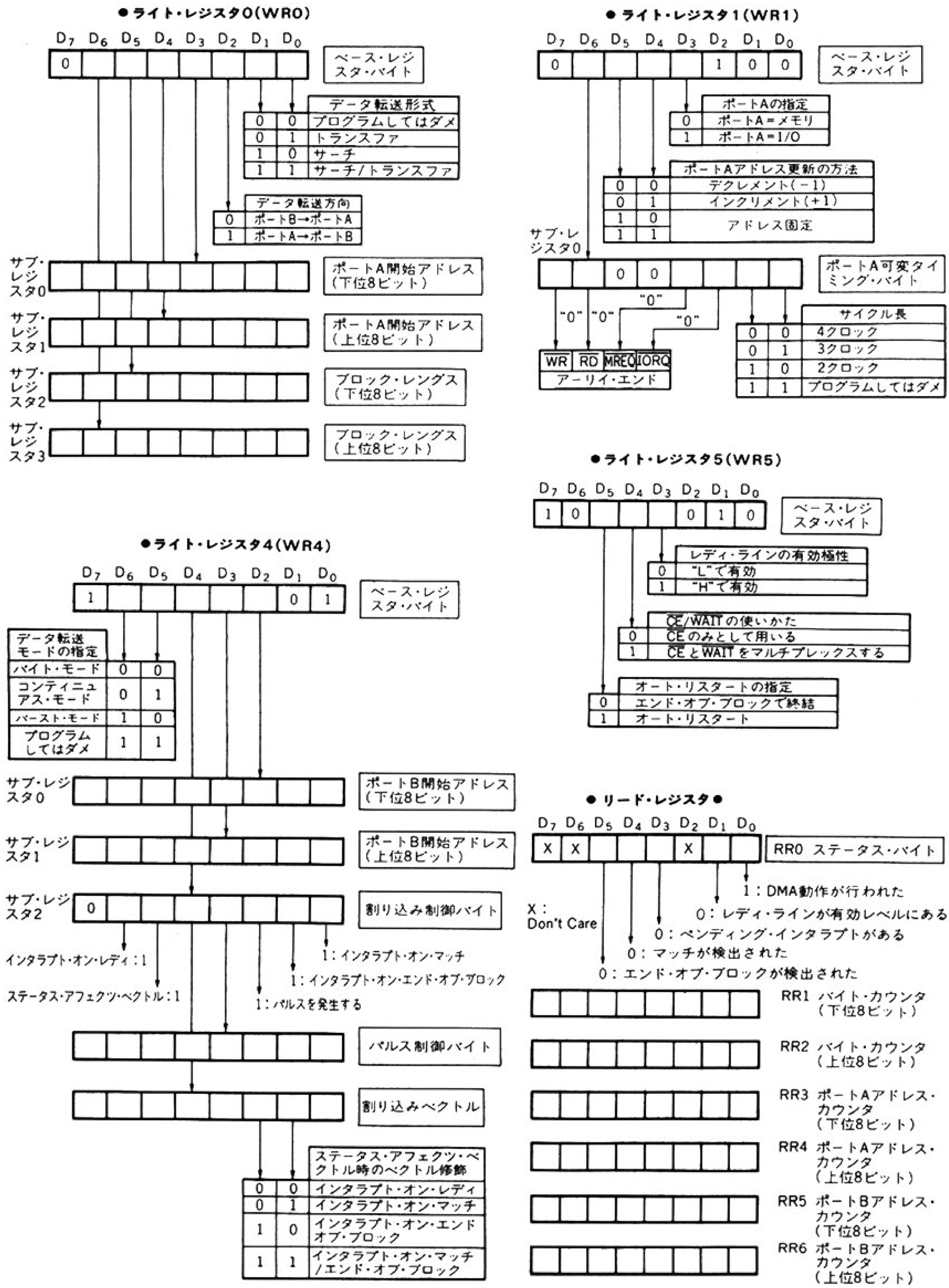
以上で「Z80 DMAのコントロール・レジスタ」の項を終わりますが，図42にZ80 DMAのライト・レジスタとリード・レジスタの構成をまとめておきます．プログラミングの際には，これらのレジスタの内容すべてを一目で見られるほうが便利だからです．

Z80 DMAは機能が豊富であるため，必然的にコマンドは相当複雑です．したがって，図41に示したイニシャライゼーション・コマンド・チェインを作るときには，よくミスをしがちです．

このミスを少しでも減らすには，アセンブラ等のマクロ機能を使うのもよい方法です．ポートA/B開始アドレス，ブロック・レングス等のパラメータをキーワード・パラメータまたはポジショナル・パラメータとして与えることにより，アセンブラは自動的にコマンド・チェインを生成してくれます．

〈図41〉 イニシャライゼーション・コマンド・チェイン．このコマンド・チェインはフル装備だが，不要のコマンド(たとえばフォース・レディ)は取り除く必要がある

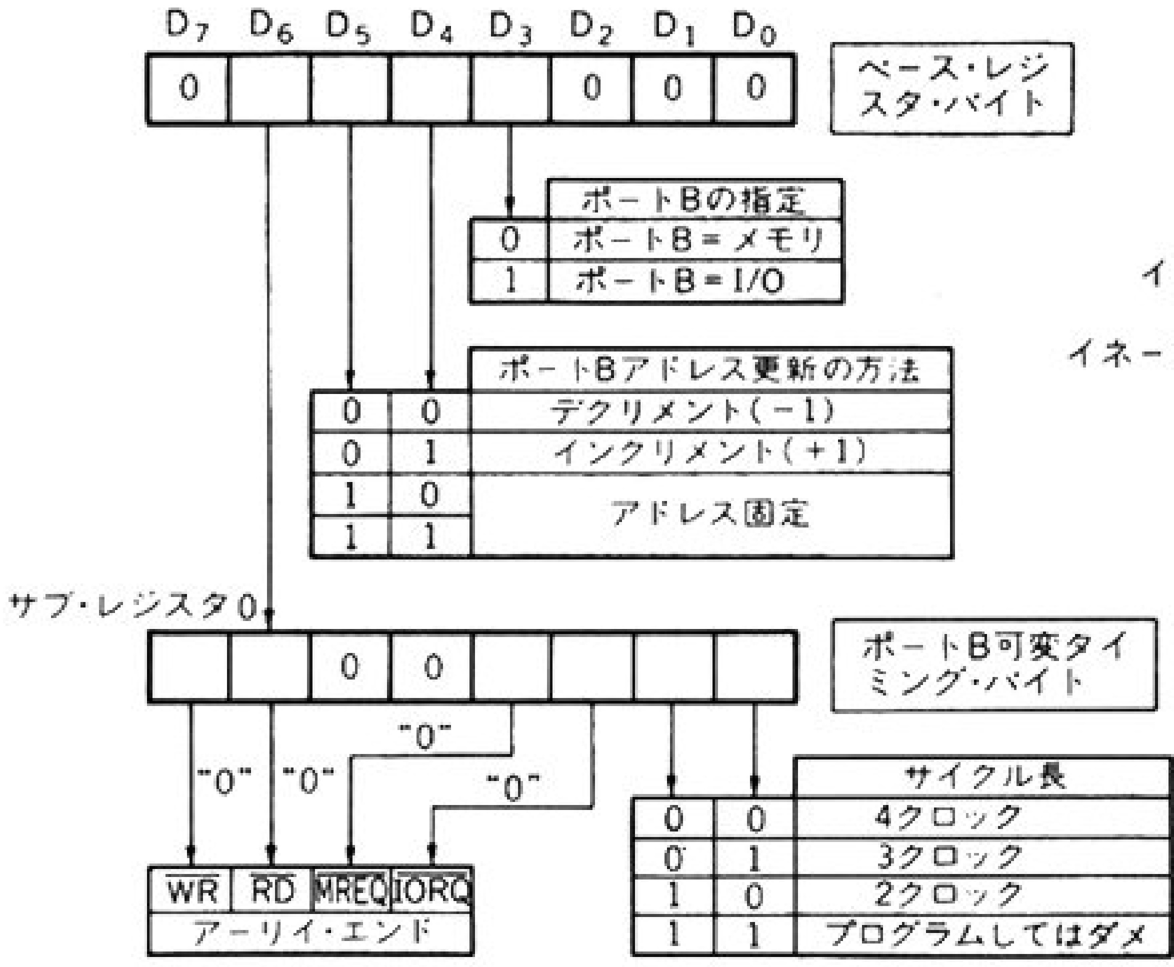
コマンド	バイト数	コマンドの内容
ディスエーブルDMAコマンド(83_{16})	1	
リセット・コマンド($C3_{16}$)	6	
WR0コマンド	5	ベース・レジスタ・バイト
		ポートA開始アドレス(下位8ビット)
		ポートA開始アドレス(上位8ビット)
		ブロック・レングス(下位8ビット)
		ブロック・レングス(上位8ビット)
WR1コマンド	2	ベース・レジスタ・バイト
		ポートA可変タイミング・バイト
WR2コマンド	2	ベース・レジスタ・バイト
		ポートB可変タイミング・バイト
WR3コマンド	3	ベース・レジスタ・バイト
		マスク・バイト
		マッチ・バイト
WR4コマンド	6	ベース・レジスタ・バイト
		ポートB開始アドレス(下位8ビット)
		ポートB開始アドレス(上位8ビット)
		割り込み制御バイト
		パルス制御バイト
		割り込みベクトル
WR5コマンド	1	ベース・レジスタ・バイト
リセット・ポートAタイミング($C7_{16}$)	1	
リセット・ポートBタイミング(CB_{16})	1	
ロード・コマンド(CF_{16})	1	
リイニシャライズ・ステータス・バイト($8B_{16}$)	1	
リード・マスク・フォローズ(BB_{16})	1	
リード・マスク	1	
イニシエイト・リード・シーケンス($A7_{16}$)	1	
フォース・レディ・コマンド($B3_{16}$)	1	
イネーブル・インタラプト(AB_{16})	1	
イネーブル・DMAコマンド(87_{16})	1	
合計バイト数	36	



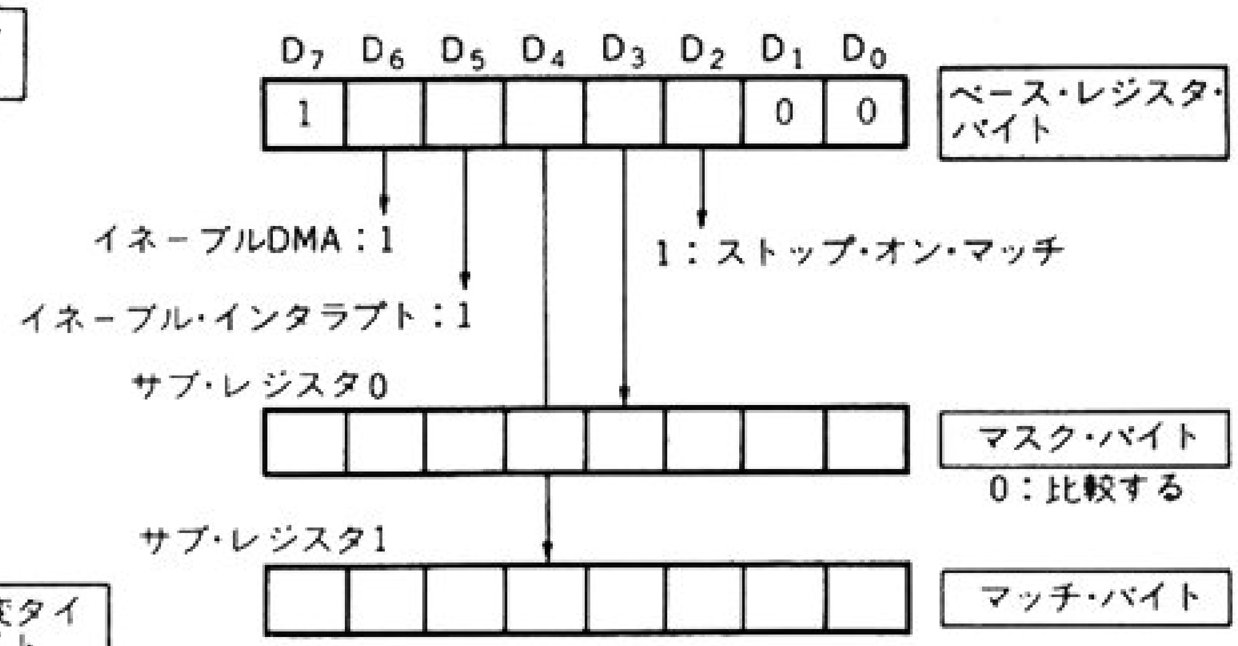
●ライト・レジスタ0(WR0)
D7 D6 D5 D4 D3 D2 D1 D0
0
ベース・レジスタ・バイト
データ転送形式
0 0 プログラムしてはダメ
0 1 トランスファ
1 0 サーチ
1 1 サーチ/トランスファ
データ転送方向
0 ポートB→ポートA
1 ポートA→ポートB
サブ・レジスタ0 ポートA開始アドレス(下位8ビット)
サブ・レジスタ1 ポートA開始アドレス(上位8ビット)
サブ・レジスタ2 ブロック・レングス(下位8ビット)
サブ・レジスタ3 ブロック・レングス(上位8ビット)
●ライト・レジスタ1(WR1)
D7 D6 D5 D4 D3 D2 D1 D0
0 1 0 0
ベース・レジスタ・バイト
ポートAの指定
0 ポートA=メモリ
1 ポートA=I/O
ポートAアドレス更新の方法
0 0 デクレメント(−1)
0 1 インクリメント(+1)
1 0 アドレス固定
1 1
サブ・レジスタ0
0 0
ポートA可変タイミング・バイト
"0" "0" "0" "0"
WR RD MREQ IORQ
アーリイ・エンド
サイクル長
0 0 4クロック
0 1 3クロック
1 0 2クロック
1 1 プログラムしてはダメ
●ライト・レジスタ5(WR5)
D7 D6 D5 D4 D3 D2 D1 D0
1 0 0 1 0
ベース・レジスタ・バイト
レディ・ラインの有効極性
0 "L"で有効
1 "H"で有効
CE/WAITの使いかた
0 CEのみとして用いる
1 CEとWAITをマルチプレックスする
オート・リスタートの指定
0 エンド・オブ・ブロックで終結
1 オート・リスタート
●ライト・レジスタ4(WR4)
D7 D6 D5 D4 D3 D2 D1 D0
1 0 1
ベース・レジスタ・バイト
データ転送モードの指定
バイト・モード 0 0
コンティニュアス・モード 0 1
バースト・モード 1 0
プログラムしてはダメ 1 1
サブ・レジスタ0 ポートB開始アドレス(下位8ビット)
サブ・レジスタ1 ポートB開始アドレス(上位8ビット)
サブ・レジスタ2 0 割り込み制御バイト
インタラプト・オン・レディ:1
ステータス・アフェクツ・ベクトル:1
1:パルスを発生する
1:インタラプト・オン・エンド・オブ・ブロック
1:インタラプト・オン・マッチ
パルス制御バイト
割り込みベクトル
ステータス・アフェクツ・ベクトル時のベクトル修飾
0 0 インタラプト・オン・レディ
0 1 インタラプト・オン・マッチ
1 0 インタラプト・オン・エンド・オブ・ブロック
1 1 インタラプト・オン・マッチ/エンド・オブ・ブロック
●リード・レジスタ●
D7 D6 D5 D4 D3 D2 D1 D0
X X X
RR0 ステータス・バイト
X: Don't Care
1:DMA動作が行われた
0:レディ・ラインが有効レベルにある
0:ペンディング・インタラプトがある
0:マッチが検出された
0:エンド・オブ・ブロックが検出された
RR1 バイト・カウンタ(下位8ビット)
RR2 バイト・カウンタ(上位8ビット)
RR3 ポートAアドレス・カウンタ(下位8ビット)
RR4 ポートAアドレス・カウンタ(上位8ビット)
RR5 ポートBアドレス・カウンタ(下位8ビット)
RR6 ポートBアドレス・カウンタ(上位8ビット)

コマンド一覧

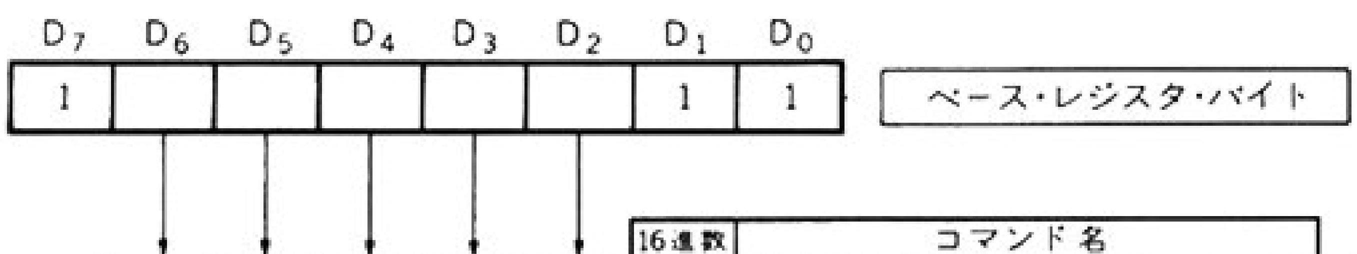
●ライト・レジスタ2(WR2)



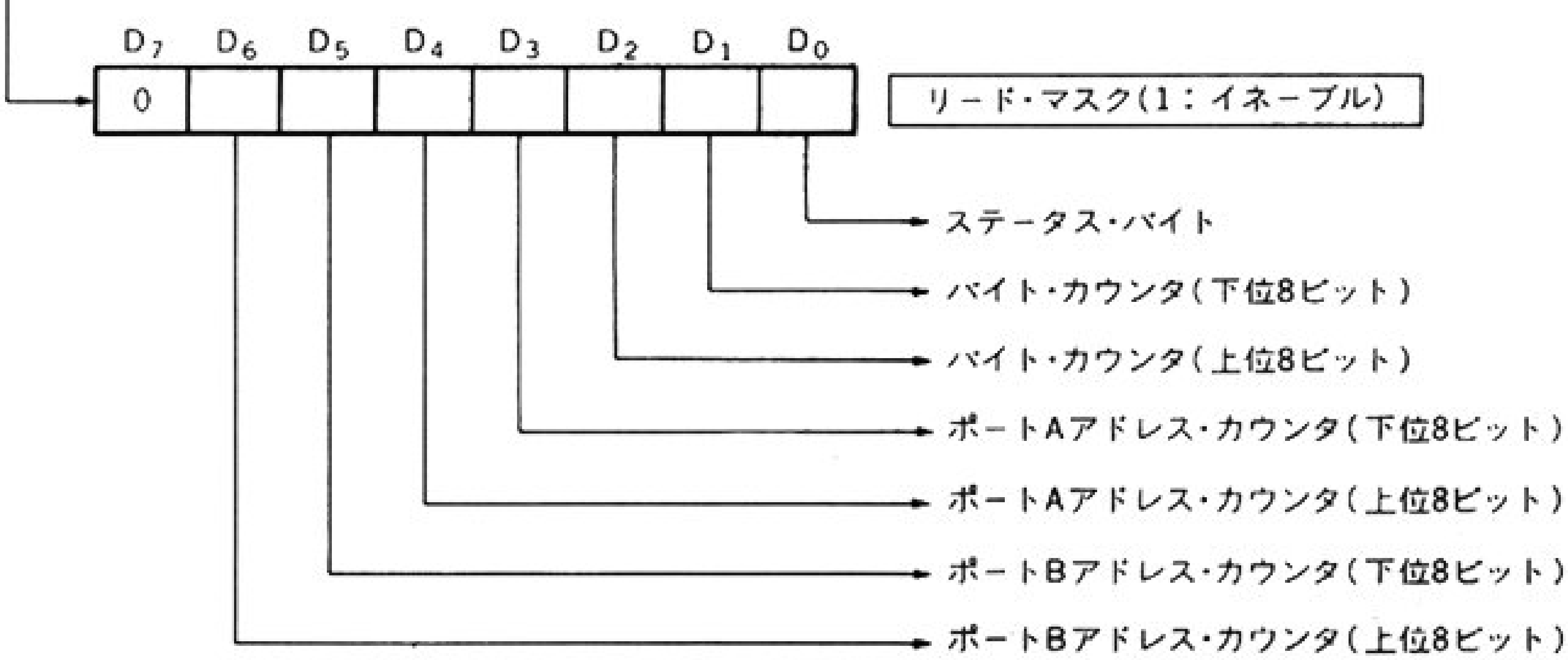
●ライト・レジスタ3(WR3)



●ライト・レジスタ6(WR6)



D6	D5	D4	D3	D2	16進数	コマンド名	
1	0	0	0	0	C3	RESET	リセット
1	0	0	0	1	C7	RESET PORT A TIMING	リセット・ポートAタイミング
1	0	0	1	0	CB	RESET PORT B TIMING	リセット・ポートBタイミング
1	0	0	1	1	CF	LOAD	ロード
1	0	1	0	0	D3	CONTINUE	コンティニュ
0	1	0	1	1	AF	DISABLE INTERRUPT	ディスエーブル・インタラプト
0	1	0	1	0	AB	ENABLE INTERRUPT	イネーブル・インタラプト
0	1	0	0	0	A3	RESET AND DISABLE INTERRUPT	リセット・アンドディスエーブル・インタラプト
0	1	1	0	1	B7	ENABLE AFTER RETI	イネーブル・アフタRETI
0	1	1	1	1	BF	READ STATUS BYTE	リード・ステータス・バイト
0	0	0	1	0	8B	REINITIALIZE STATUS BYTE	リイニシャライズ・ステータス・バイト
0	1	0	0	1	A7	INITIATE READ SEQUENCE	イニシエイト・リード・シーケンス
0	1	1	0	0	B3	FORCE READY	フォース・レディ
0	0	0	0	1	87	ENABLE DMA	イネーブルDMA
0	0	0	0	0	83	DISABLE DMA	ディスエーブルDMA
0	1	1	1	0	BB	READ MASK FOLLOWS	リード・マスク・フォローズ



●Z80ファミリとインテル系の周辺LSI●

周辺LSIの系列は昔から3本の流れ，つまりインテル系8080系とモトローラの6800系，それにザイログのZ80系として走ってきました．Z80 CPUを用いる場合，さすがに6800系の周辺LSIを使うことは，CRTC 6845を除いてめったにありませんが，インテル系の周辺LSIを使うことはよくあります．したがって，Z80ファミリと8080系の周辺LSIとの相違を知っていてもムダではありません．

8080系の周辺LSI，というよりマイクロプロセッサの周辺LSIの歴史はインテルの8255(PPI)，8251(USART)，8253(PIT)ならびに8257(DMAC)により始まったといっても過言ではありません。これらのLSIは現在なお健在であり，8086，8088を用いた16ビット・システムでも多用されています．

ただし，8251はSDLCの制御ができませんので，この機能を用いるときには8274MPCCが使われます．また，8257もやや機能不足の感があり，現在では8237が使われることがほとんどです．この8237は，AMD社が8257を改良して作ったAM9517のセカンド・ソースです．

また，比較表には示しませんでしたが，インテル系には8259 PICというインタラプト・コントローラがあります．インテル系の周辺LSIには割り込みベクトルを発生する機能がありませんので，各LSIの割り込み要求線をこの8259に集め，8259が優先順位制御をし，割り込みベクトルを発生します．

Z80系の周辺LSIも表に示したものだけでなく，Z8090 UPC (Universal Peripheral Controller)，Z8038 FIFO，Z8530 SCC (Serial Communication Controller)，Z8536 CIO(Counter/Timer & Parallel I/O)等と発展しており，これらはZ8，Z80，Z800，Z8000等のザイログ社のCPUだけではなく，他のCPUとも容易にインターフェースできるように設計されたユニバーサル・ペリフェラルです．

Z80ファミリとインテル系の周辺LSIの比較表

パラレル・インターフェース		
比較項目	Z80 PIO	8255 PPI
ピンの数	40	40
ポート数	2	3
モード	0：出力 1：入力 2：双方向性 3：ビット・モード	0：基本入出力 1：ストローブ付き入出力 2：双方向性
カウンタ・タイマ・サーキット		
比較項目	Z80 CTC	8253 PIT
ピンの数	28	24
カウンタの数	4	3
モード	タイマ・モード カウンタ・モード	インタラプト・オンTC プログラマブル・ワンショット レート・ジェネレータ 方形波発生 ソフトウェア・トリガ ハードウェア・トリガ
カウンタの幅	8ビット(ただし16/256のプリスケーラ付き)	16ビット
シリアル・インターフェース		
比較項目	Z80 SIO	8251 USART
ピンの数	40	28
ポート数	2	1
モード	非同期 同期 SDLC	非同期 同期
速　度	880KBPS	19.2KBPS(非同期) 64KBPS(同期)
DMAコントローラ		
比較項目	Z80 DMA	8257 DMAC
ピンの数	40	40
チャネル数	1	4
転送速度	2Mバイト/秒	750Kバイト/秒
アドレス・カウンタ	16ビット	16ビット
レングス・カウンタ	16ビット	14ビット
その他	サーチ機能 メモリ↔メモリ転送 I/O↔I/O転送 パルス出力機能	ローテイティング・プライオリティ

Z80 DMAのタイミング

以下の説明では，Z80 DMAがDMAデータ転送をしているとき($\overline{BUSACK}$＝"H")のことを「CPUがバス・マスタ(Bus Master)である」と表現し，Z80 DMAがDMAデータ転送をしているとき($\overline{BUSACK}$＝"L")のことを「DMAがバス・マスタである」と呼びます．

バス・マスタということばは，バスの主人のことであり，バスの制御権を持っているものという意味です．

CPUがバス・マスタであるとき

このときには，Z80 DMAはCPUからコマンド(ライト・レジスタに対する書き込み)を受け付けたり，リード・レジスタに対する読み取り動作に応じます．

■ライト・レジスタへの書き込みタイミング

図43にCPUがバス・マスタであるときの，ライト・レジスタへの書き込みタイミングを示します．ライト・レジスタにアクセスするには，アドレス・デコーダの出力である$\overline{CE}$(チップ・イネーブル)と，$\overline{IORQ}$および$\overline{WR}$を"L"にします．

Z80 DMAはこれら3本の信号がすべて"L"になると，次のクロックの立ち上がり時にこの状態をラッチし，その次のクロックの立ち上がり時にデータ・バス($D_0 \sim D_7$)の状態を必要なライト・レジスタに書き込みます．

したがって，$\overline{CE}$，$\overline{IORQ}$，ならびに$\overline{WR}$信号は，Z80 DMAにラッチされた後，ある一定のホールド時間後には無効レベルに変化してもかまいません．

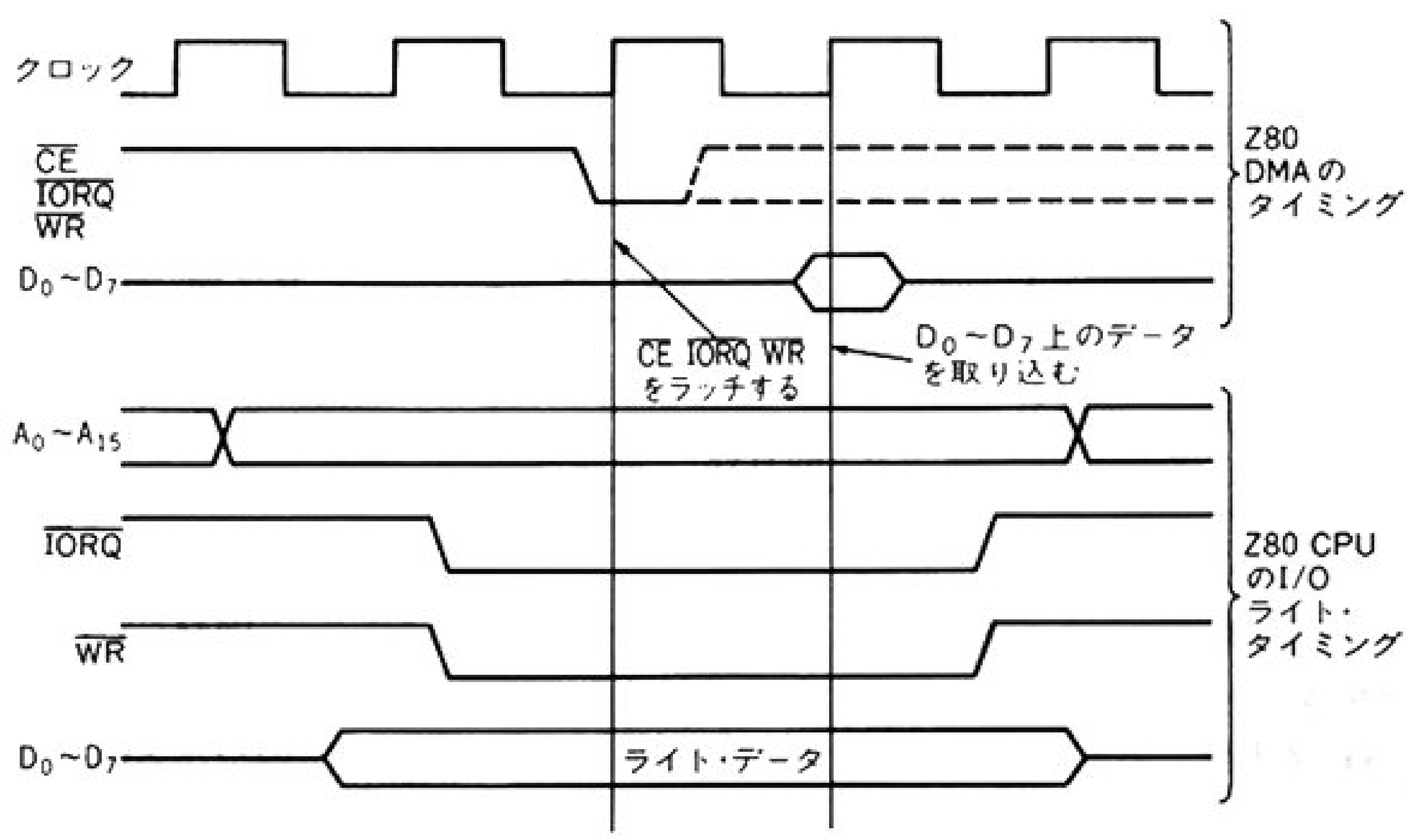
Z80 DMAがZ80 CPUのタイミングで動作するのは当然ですが，参考のため**図43**にはZ80 CPUのI/Oライト・タイミングも示してあります．この図から，Z80 DMAのライト・レジスタへのアクセスは，Z80 CPUのI/Oライト・タイミングで十分すぎることがわかります．

■リード・レジスタからの読み出しタイミング

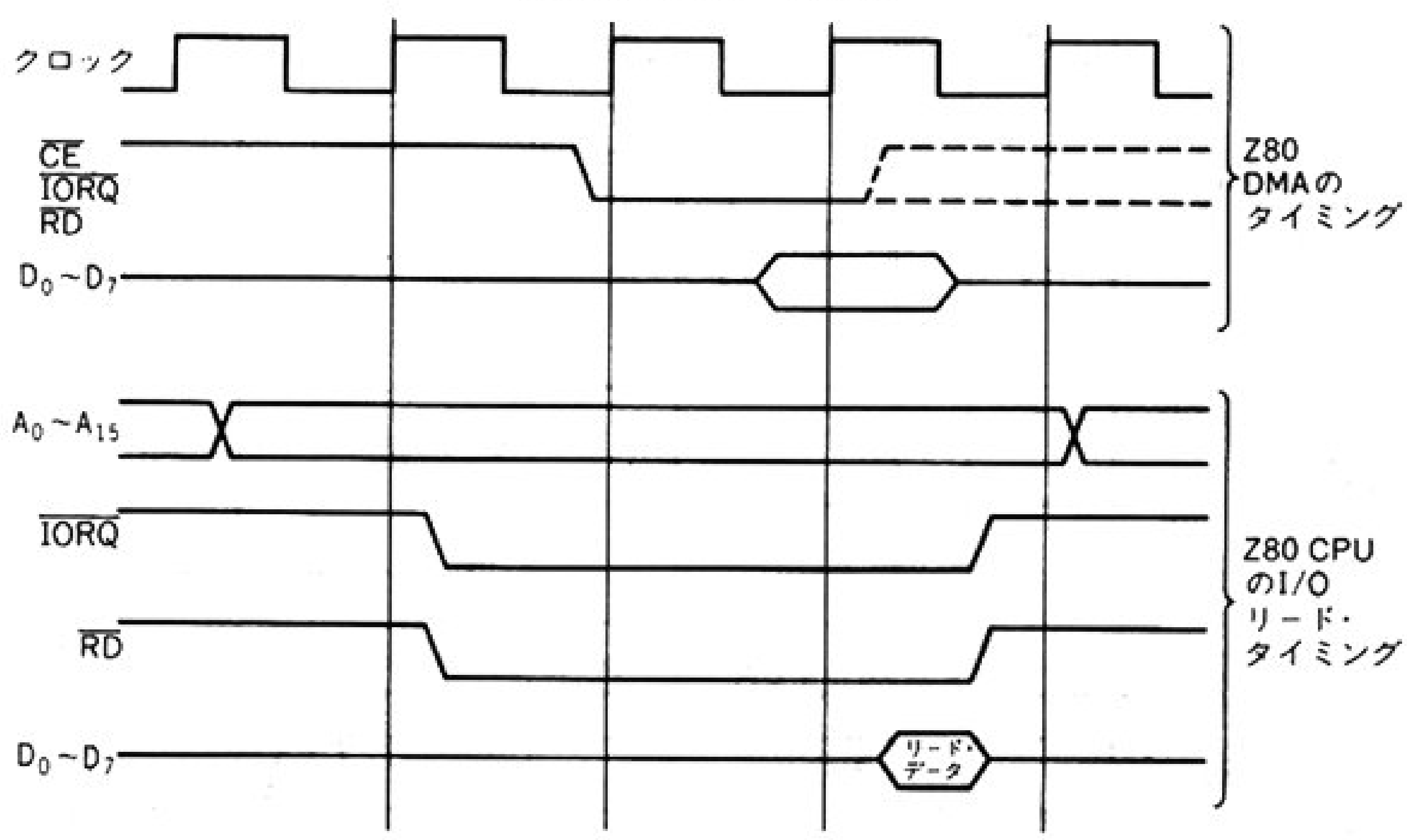
図44にCPUがバス・マスタであるときの，Z80 DMAのリード・レジスタに対する読み出しタイミングを示します．リード・レジスタにアクセスするには，$\overline{CE}$，$\overline{IORQ}$ならびに$\overline{RD}$を"L"にします．

Z80 DMAはこれら3本の信号が"L"になるのを検出すると，データ・バス上にリード・レジスタの内容を乗せます．$\overline{CE}$，$\overline{IORQ}$，$\overline{RD}$を同時に"L"にしておく時間は，ザイログ社のデータ・シートには記されていませんので，正確にはわかりませんが，3本の信号が"L"になってからデータがバス上に乗せられるまで

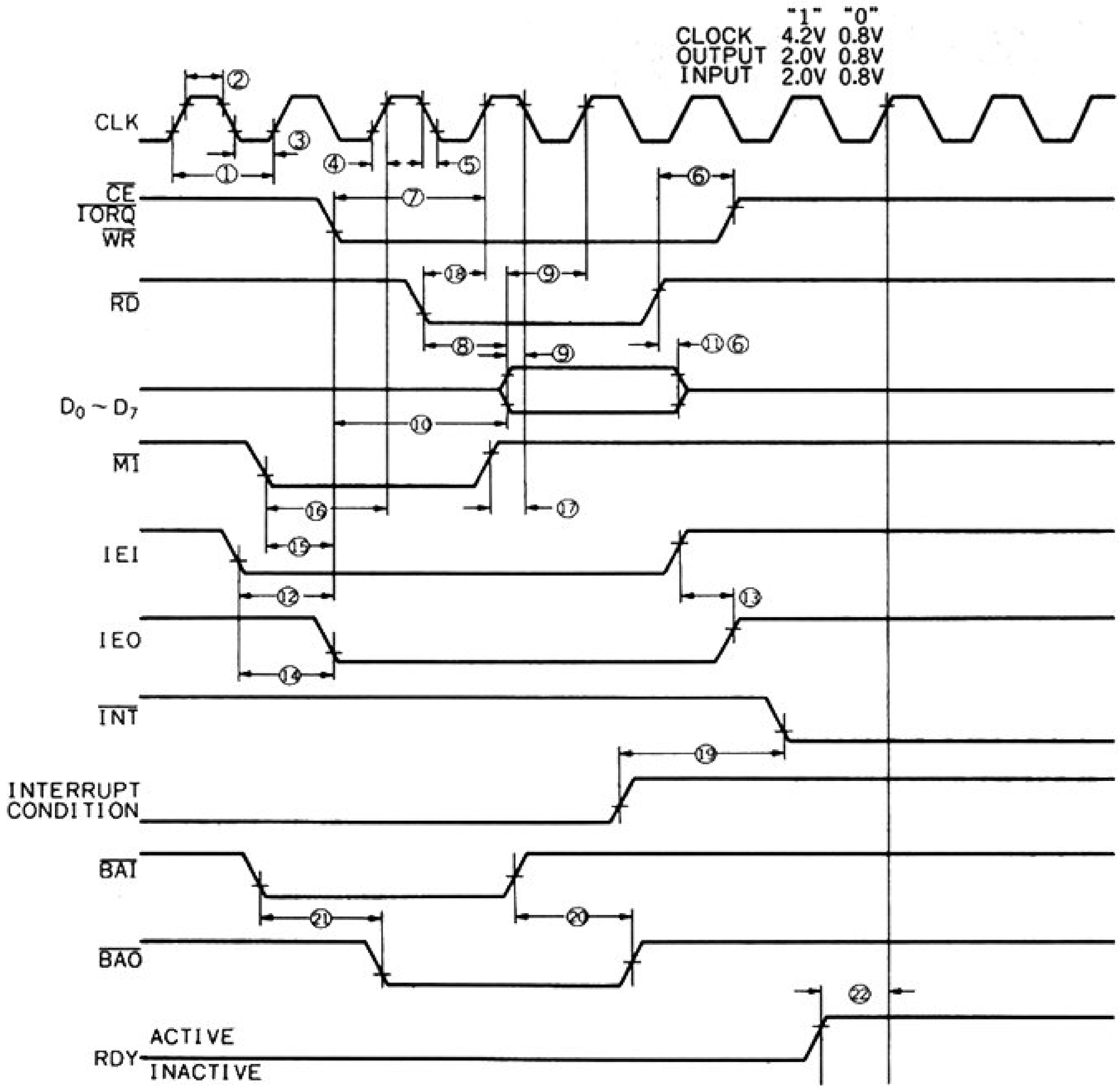
〈図43〉 Z80 DMAのライト・レジスタへの書き込みタイミング(CPUがバス・マスタ)



〈図44〉 Z80 DMAのリード・レジスタに対する読み出しタイミング(CPUがバス・マスタ)



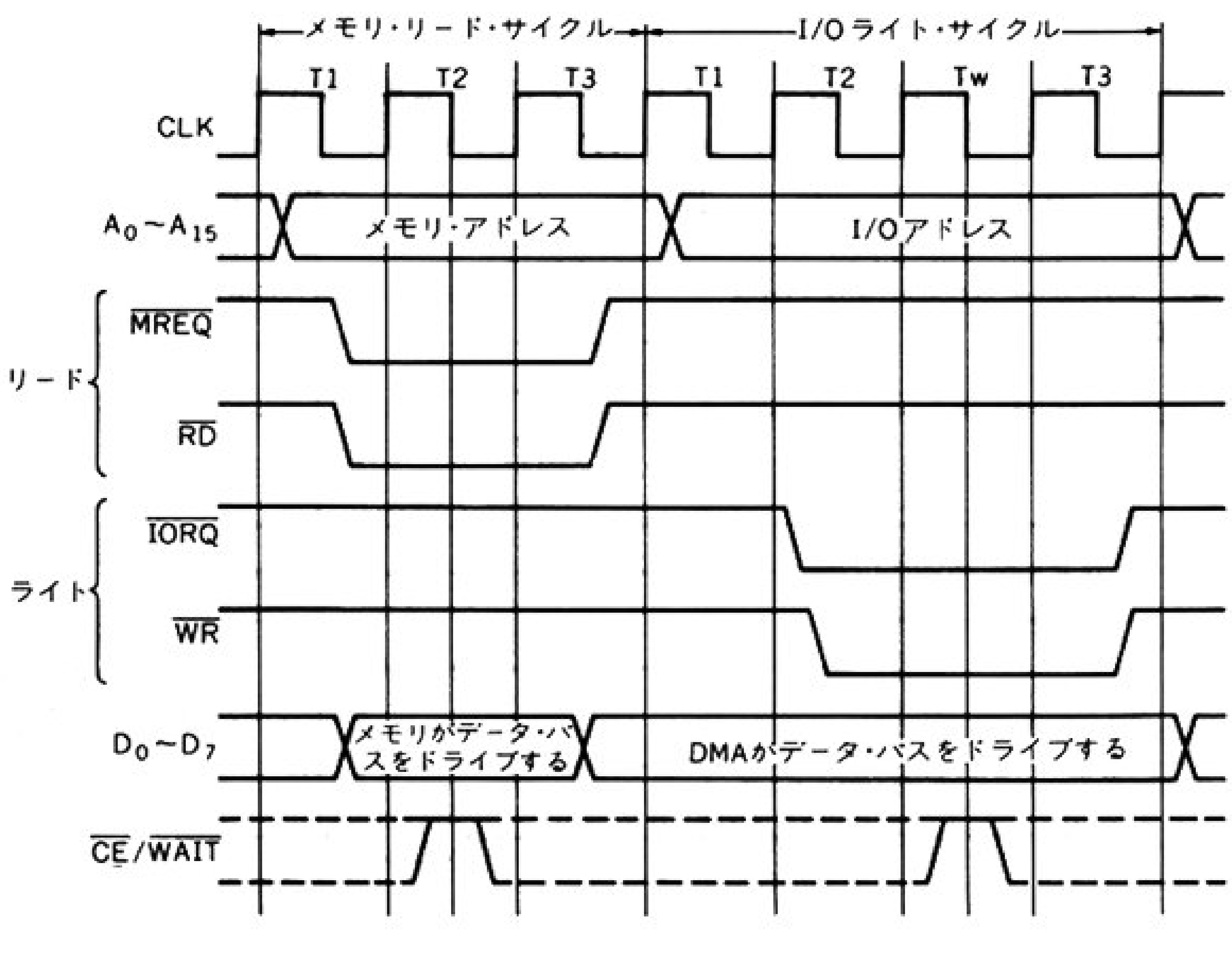
〈図45〉
Z80 DMA のタイミング(CPUがバス・マスタであるとき)



の時間（$T_{dDO(RDf)}$）の最大値は，Z80 DMAでは500ns，Z80A DMAでは380nsですので，1.5クロック期間もあれば十分であると思われます。

図44にはZ80 CPUのI/Oリード・タイミングも示してありますが，Z80 CPUの$\overline{\mathrm{IORQ}}$，および$\overline{\mathrm{RD}}$信号のパルス幅は2クロック期間以上ありますので，もちろんのことですが問題ありません。

図45にZ80 DMA，およびZ80A DMAの，CPUがバス・マスタにあるときの正確なタイミング・チャートを示します。



〈図46〉
DMAデータ転送タイミング
(メモリ → I/O)

Z80 DMAがバス・マスタであるとき

ここではZ80 DMAがバス・マスタにあるとき，つまりZ80 DMA本来の使命であるDMAデータ転送を行っているときのタイミングについ

番号	記号	パラメータ	Z80 DMA Min	Z80 DMA Max	Z80A DMA Min	Z80A DMA Max	単位
1	TcC	Clock Cycle Time	400	4000	250	4000	ns
2	TwCh	Clock Width (High)	170	2000	110	2000	ns
3	TwCl	Clock Width (Low)	170	2000	110	2000	ns
4	TrC	Clock Rise Time		30		30	ns
5	TfC	Clock Fall Time		30		30	ns
6	Th	Hold Time for Any Specified Setup Time	0		0		ns
7	TsC(Cr)	$\overline{\text{IORQ}}$, $\overline{\text{WR}}$, $\overline{\text{CE}}$ ↓ to Clock ↑ Setup	280		145		ns
8	TdDO(RDf)	$\overline{\text{RD}}$ ↓ to Data Output Delay		500		380	ns
9	TsWM(Cr)	Data In to Clock ↑ Setup ($\overline{\text{WR}}$ or $\overline{\text{M1}}$)	50		50		ns
10	TdCf(DO)	$\overline{\text{IORQ}}$ ↓ to Data Out Delay (INTA Cycle)		340		160	ns
11	TdRD(Dz)	$\overline{\text{RD}}$ ↑ to Data Float Delay (output buffer disable)		160		110	ns
12	TsIEI(IORQ)	IEI ↓ to $\overline{\text{IORQ}}$ ↓ Setup (INTA Cycle)	140		140		ns
13	TdIEOr(IEIr)	IEI ↑ to IEO ↑ Delay		210		160	ns
14	TdIEOf(IEIf)	IEI ↓ to IEO ↓ Delay		190		130	ns
15	TdM1(IEO)	$\overline{\text{M1}}$ ↓ to IEO ↓ Delay (interrupt just prior to $\overline{\text{M1}}$ ↓)		300		190	ns
16	TsM1f(Cr)	$\overline{\text{M1}}$ ↓ to Clock ↑ Setup	210		90		ns
17	TsM1r(Cf)	$\overline{\text{M1}}$ ↑ to Clock ↓ Setup	20		−10		ns
18	TsRD(Cr)	$\overline{\text{RD}}$ ↓ to Clock ↑ Setup ($\overline{\text{M1}}$ Cycle)	240		115		ns
19	TdI(INT)	Interrupt Cause to $\overline{\text{INT}}$ ↓ Delay ($\overline{\text{INT}}$ generated only when DMA is inactive)		500		500	ns
20	TdBAIr(BAOr)	$\overline{\text{BAI}}$ ↑ to $\overline{\text{BAO}}$ ↑ Delay		200		150	ns
21	TdBAIf(BAOf)	$\overline{\text{BAI}}$ ↓ to $\overline{\text{BAO}}$ ↓ Delay		200		150	ns
22	TsRDY(Cr)	RDY Active to Clock ↑ Setup	150		100		ns

て説明します．なお，ここではZ80標準タイミングの場合だけ述べ，可変タイミングについては後述します．

■メモリ → I/O

図46にメモリからI/Oへの方向のDMAデータ転送タイミングを示します．この図のタイミングは，トランスファおよびトランスファ／サーチの両方に適用されます．

この方向のデータ転送では，Z80 DMAはまずアドレス・バス(A_0～A_{15})上にメモリ・アドレスを乗せ，ついで$\overline{\text{MREQ}}$および$\overline{\text{RD}}$を"L"にします．メモリから読み出されたデータ・バス(D_0～D_7)上のデータは，$\overline{\text{RD}}$信号が立ち上がる直前のクロック変化時（標準タイミングではT3の立ち下がり時）に，Z80 DMA内部にラッチされます．そして$\overline{\text{RD}}$信号が"H"になると，Z80 DMAのデータ・バス・バッファがイネーブルされラッチされていたデータが，データ・バス上に出力されます．

次にZ80 DMAはI/Oライト・サイクルに入りますが，まずI/Oアドレスをアドレス・バス上に乗せ，次に$\overline{\text{IORQ}}$，および$\overline{\text{WR}}$を"L"にして，データ・バス上のデータ（メモリから読み出されたデータ）をI/Oに書き込みます．

$\overline{\text{CE}}$/$\overline{\text{WAIT}}$ラインが"マルチプレックス"とプログラミングされているときには(WR5)，Z80 DMAは，メモリ・リードのときにはT2の立ち下がり時点，I/OライトのときにはTwの立ち下がり時点に，同ラインの状態をサンプリングします．

このとき，$\overline{\text{WAIT}}$信号が"L"であればZ80 DMAは1クロック期間ウェイト・サイクル(Tw)を挿入し，Twの立ち下がり時に再度$\overline{\text{CE}}$/$\overline{\text{WAIT}}$ラインの状態をチェックします．このとき$\overline{\text{WAIT}}$ラインの状態が"L"であれば，さらに1ウェイト・サイクルを挿入し，"H"であれば，次のサイクルへ進みます．

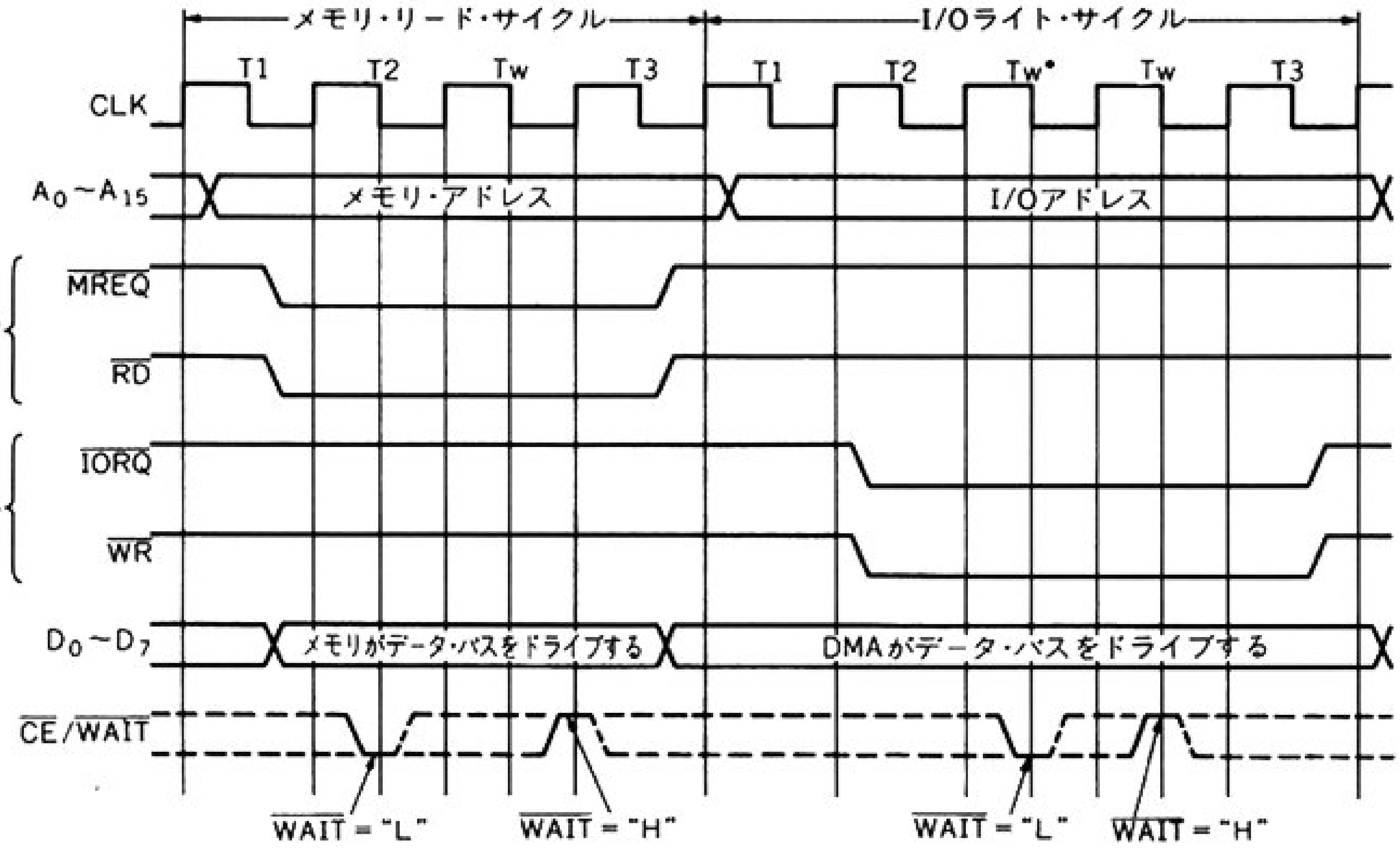
図47に，1ウェイト・サイクルを挿入した場合の，メモリ → I/Oのタイミングを示します．

■I/O → メモリ

図48にI/Oからメモリへの向きのDMAデータ転送タイミングを示します．この場合には，Z80 DMAはまずアドレス・バス上にI/Oアドレスを乗せ，ついで$\overline{\text{IORQ}}$，および$\overline{\text{RD}}$を"L"にします．

I/Oがデータ・バス上にデータを乗せますと，Z80

〈図47〉
DMA データ転送タイミング(メモリ → I/O), 1ウェイト・サイクルを挿入したとき



Tw*：自動的に挿入されるウェイト・サイクル　　Tw：WAITにより挿入されるウェイト・サイクル

DMA は $\overline{RD}$ の立ち上がりの直前のクロックの立ち下がり時点（標準タイミングでは T3 の立ち下がり時）にそのデータを内部にラッチし，$\overline{RD}$ が "H" になるとそのデータをデータ・バス上に乗せます．

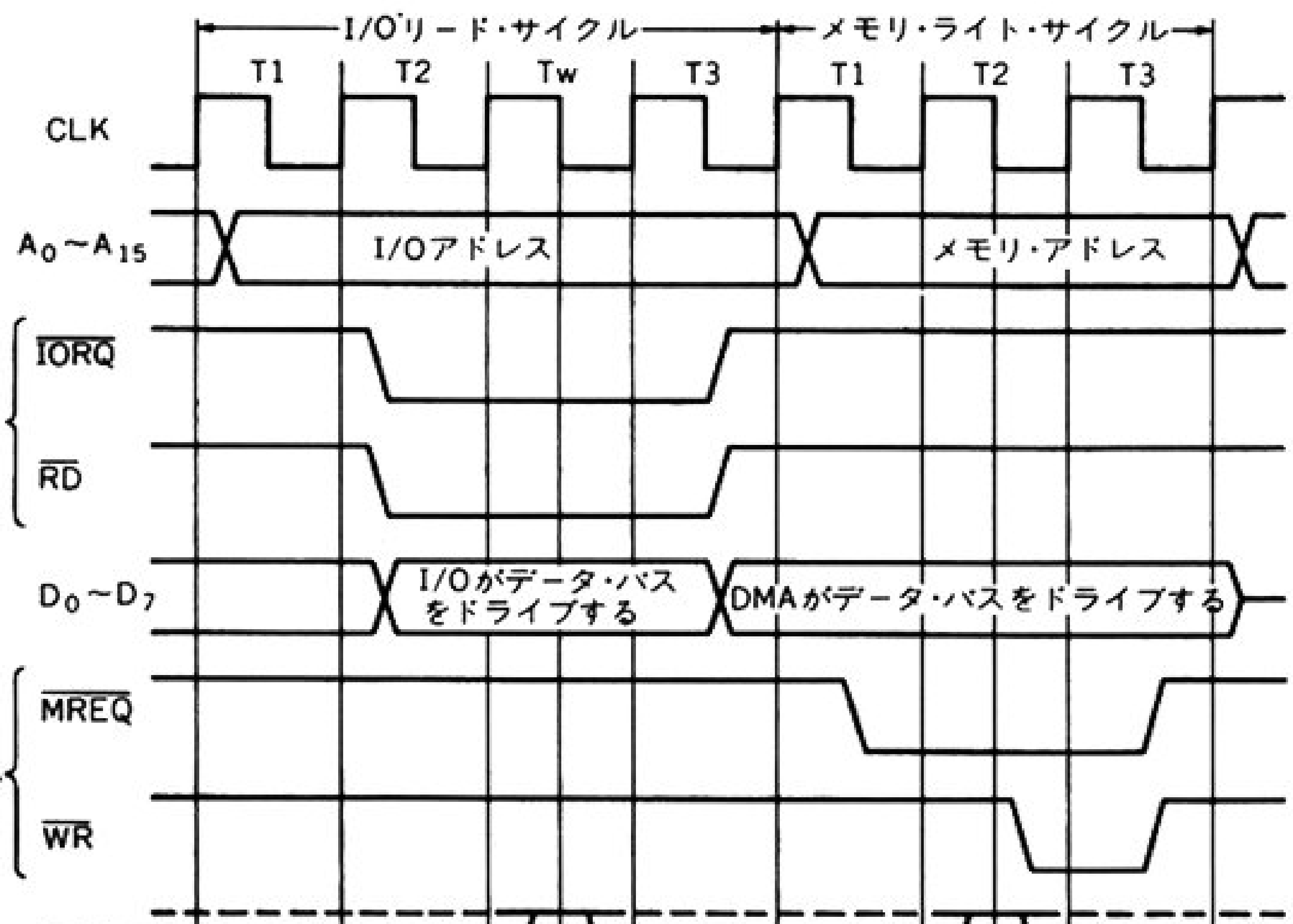
次に Z80 DMA はメモリ・ライト・サイクルに入り，アドレス・バス上にメモリ・アドレスを乗せ，ついで $\overline{MREQ}$，および $\overline{WR}$ を "L" にして，I/O から読み取られたデータをメモリに書き込みます．

$\overline{CE}/\overline{WAIT}$ ラインの取り扱いは，メモリ → I/O の場合と同じで，I/O リードのときには Tw の立ち下がり時に，メモリ・ライトのときには T2 の立ち下がり時にサンプリングします．図49に 1 Tw ステートを挿入した場合の，I/O → メモリのタイミングを示します．

■メモリ → メモリ

図50にメモリ → メモリの DMA データ転送タイミングを示します．この方向のデータ転送タイミングは，図46（メモリ → I/O）のメモリ・リード・サイクルと図48（I/O → メモリ）のメモリ・ライト・サイクルをくっつけたものです．したがってとくに説明の必要はないと思います．

〈図48〉 **DMA データ転送タイミング（I/O → メモリ）**



■I/O → I/O

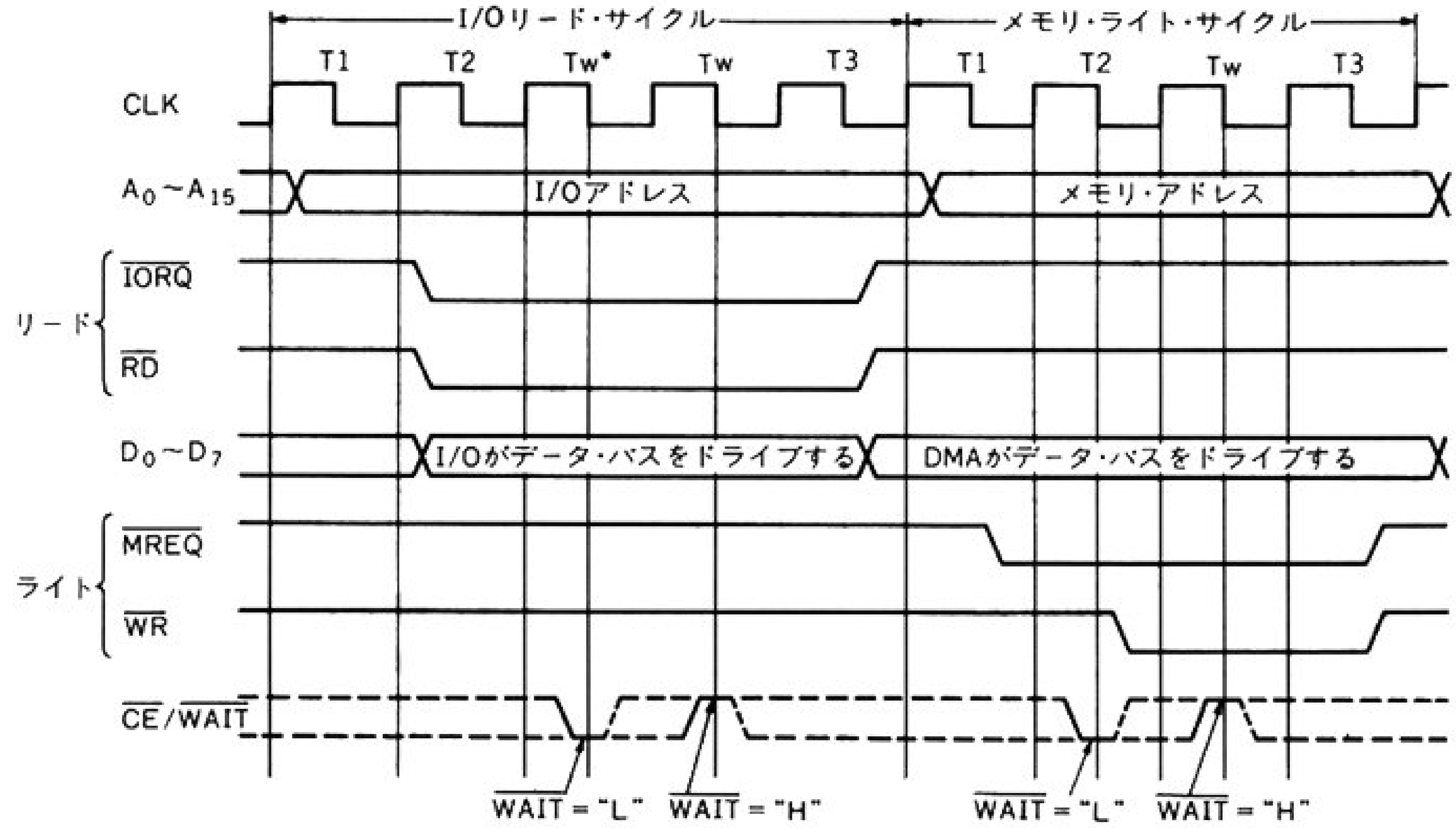
図51に I/O → I/O の DMA データ転送タイミングを示します．このタイミングは図48の I/O リード・サイクルと図46の I/O ライト・サイクルをくっつけたものです．

■サーチ・タイミング

サーチ・メモリ，およびサーチ I/O のタイミングは，それぞれメモリ → I/O，および I/O → メモリのタイミング中のリード・サイクルが該当します．したがって，サーチ・メモリの場合は図46のメモリ・リード・タイミングがそのまま当てはまり，サーチ I/O の場合は，図48中の I/O リード・サイクルが当てはまります．

＊

以上の説明から，Z80 DMA を用



〈図49〉
DMAデータ転送タイミング(I/O → メモリ)，1ウェイト・サイクルを挿入したとき

Tw*：自動的に挿入されるウェイト・サイクル　　Tw：WAITにより挿入されるウェイト・サイクル

いたDMAデータ転送タイミングは，可変タイミングを用いない逐次データ転送を用いる限り，Z80 CPUのリード／ライト・タイミングと同じです．したがって，メモリやI/Oのリード／ライト・タイミングを設計する際に，DMAデータ転送をするからといって，とくに考慮する必要はありません．

もちろん，可変タイミングを用いたり，次項で説明する同時データ転送を行うときには，この限りではありません．また Z80 DMA は，DMA データ転送中にリフレッシュ信号を出す機能がありませんので，Z80 CPU の $\overline{\text{RFSH}}$ 信号を用いてダイナミック RAM のリフレッシュを行っており，かつ相当長い間，バースト・モードまたはコンティニュアス・モードでDMAデータ転送を行うときには，別のリフレッシュ方法を講じる必要があります．

同時データ転送タイミング（CPUがバス・マスタ）

前に述べましたように，同時データ転送では，Z80 DMA にメモリ・サーチ，またはI/Oサーチをやらせておき，外部ロジックによりメモリ，およびI/Oの制御信号を発生し，1リード・サイクル中にデータのリード／ライトを行ってしまいます．以下に示すタイミング中の，(EXT)を付けた信号は外部ロジックが発生する信号です．

■メモリ → I/O（メモリ・サーチ・タイミング）

図52に Z80 DMAのメモリ・サーチ・タイミングにおいて，メモリ → I/O のデータ転送を行うタイミングを示します．

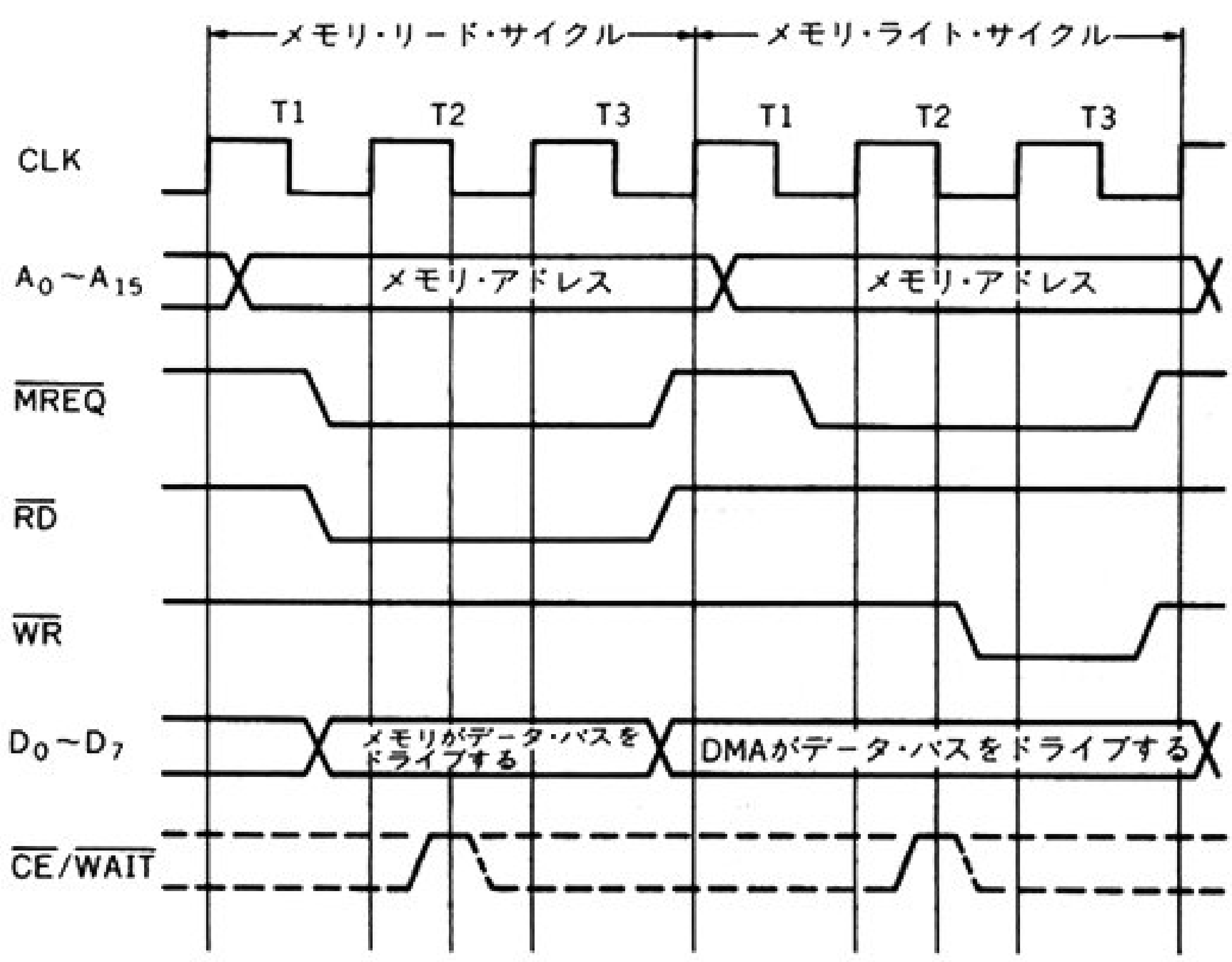
この場合には，メモリ・リード・タイミングに，$\overline{\text{IORQ}}$ および $\overline{\text{WR}}$ を外部ロジックにより発生します．

■I/O → メモリ（メモリ・サーチ・タイミング）

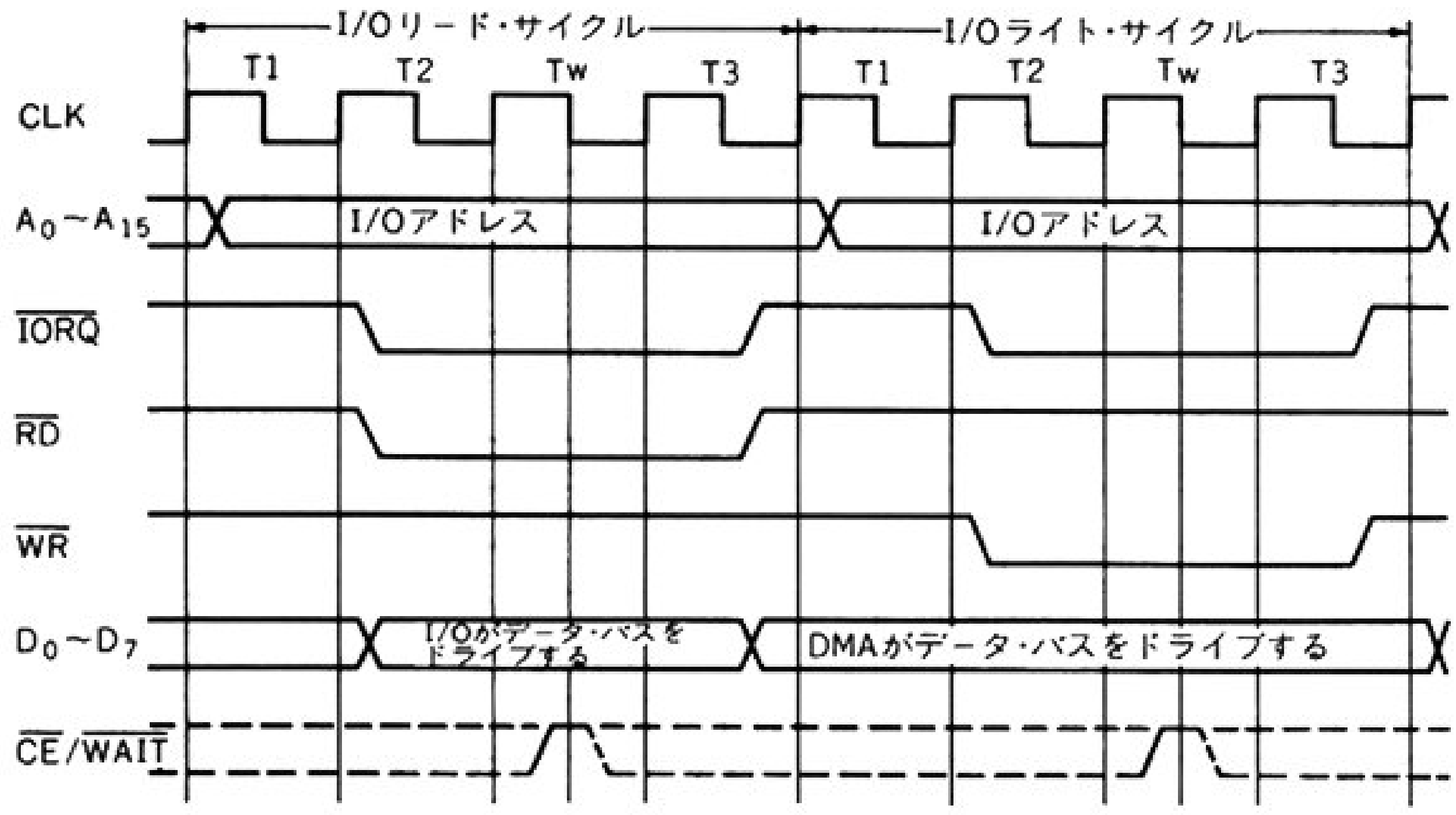
図53に Z80 DMAのメモリ・サーチ・タイミングにおいて，I/O → メモリのデータ転送を行うタイミングを示します．

この場合には，メモリ・リード・タイミングに，$\overline{\text{IORQ}}$ および $\overline{\text{WR}}$ 信号を外部ロジックにより発生し， Z80 DMAが出力する $\overline{\text{RD}}$ と，外部ロジックが発生する $\overline{\text{IORQ}}$

〈図50〉 **DMAデータ転送タイミング（メモリ → メモリ）**



〈図51〉 DMAデータ転送タイミング (I/O → I/O)



により，I/O リードを行い，Z80 DMA が出力する $\overline{\text{MREQ}}$ と外部ロジックが発生する $\overline{\text{WR}}$ により，メモリ・ライトを行います．

■メモリ → I/O（I/O サーチ・タイミング）

図54に Z80 DMA の I/O サーチ・タイミングを用いた，メモリ → I/O のデータ転送タイミングを示します．

この場合には I/O リードのタイミングに，外部ロジックにより $\overline{\text{MREQ}}$ および $\overline{\text{WR}}$ を発生し，Z80 DMA が出力する $\overline{\text{RD}}$ と外部ロジックが発生する $\overline{\text{MREQ}}$ により，メモリ・リードを行い，外部ロジックが発生する $\overline{\text{WR}}$ と Z80 DMA が出力する $\overline{\text{IORQ}}$ とにより，I/O ライトを行います．

■I/O → メモリ（I/O サーチ・タイミング）

図55に I/O サーチ・タイミングを用いた，I/O → メモリのデータ転送タイミングを示します．この場合には I/O リード・タイミングに，外部ロジックにより，$\overline{\text{MREQ}}$ および $\overline{\text{WR}}$ を発生し，Z80 DMA が I/O リードを行うと同時にメモリ・ライトを行います．

*

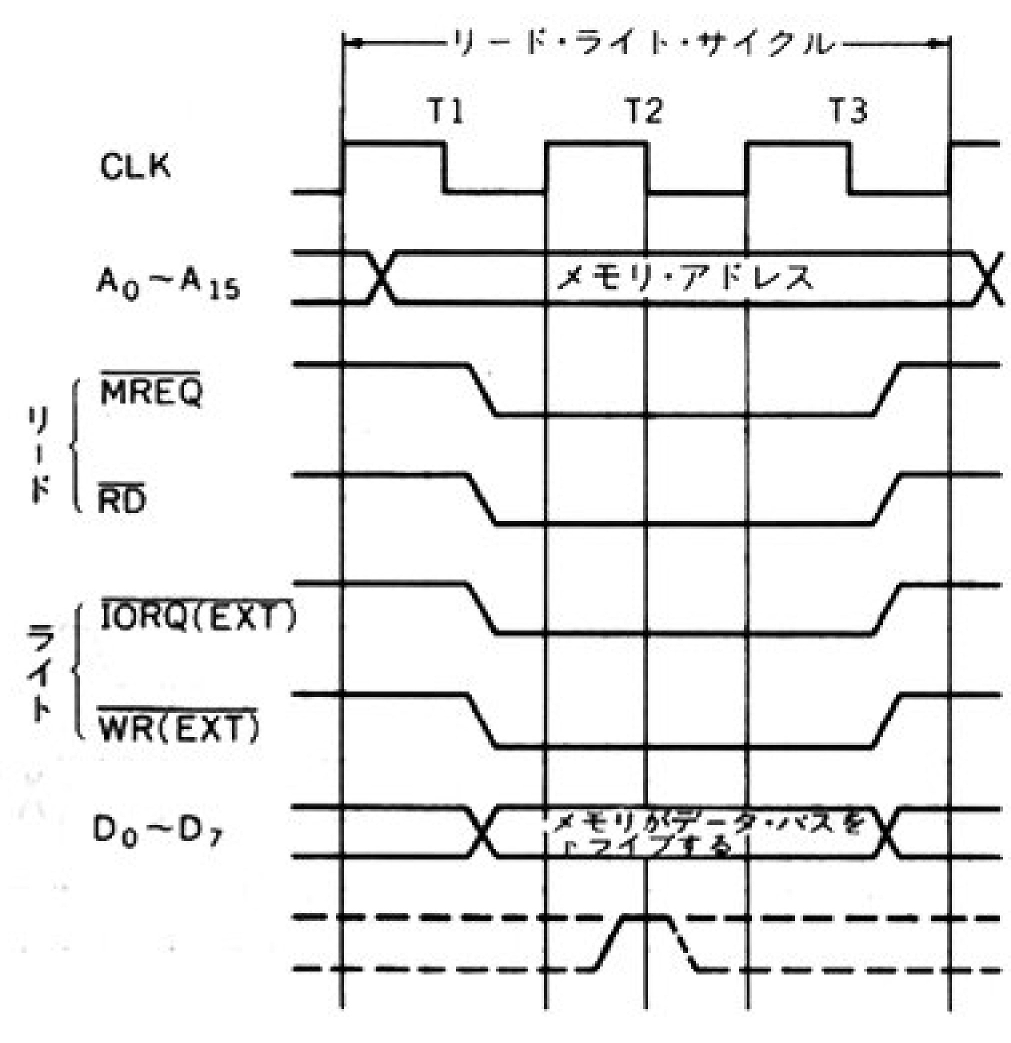
以上，四つのタイミングすべてにおいて，Z80 DMA が出力するアドレスは，メモリ・アドレスであると解釈し，I/O のセレクトは外部ロジックにより行います．

これは DMA データ転送時に，I/O アドレスは通常固定しており，メモリ・アドレスは変化するのがふつうですので，メモリ・アドレスの管理を Z80 DMA にまかせておいたほうが有利だからです．したがって I/O アドレスはどこからも出力されませんので，外部ロジックにより I/O のセレクト信号を出す必要があります．

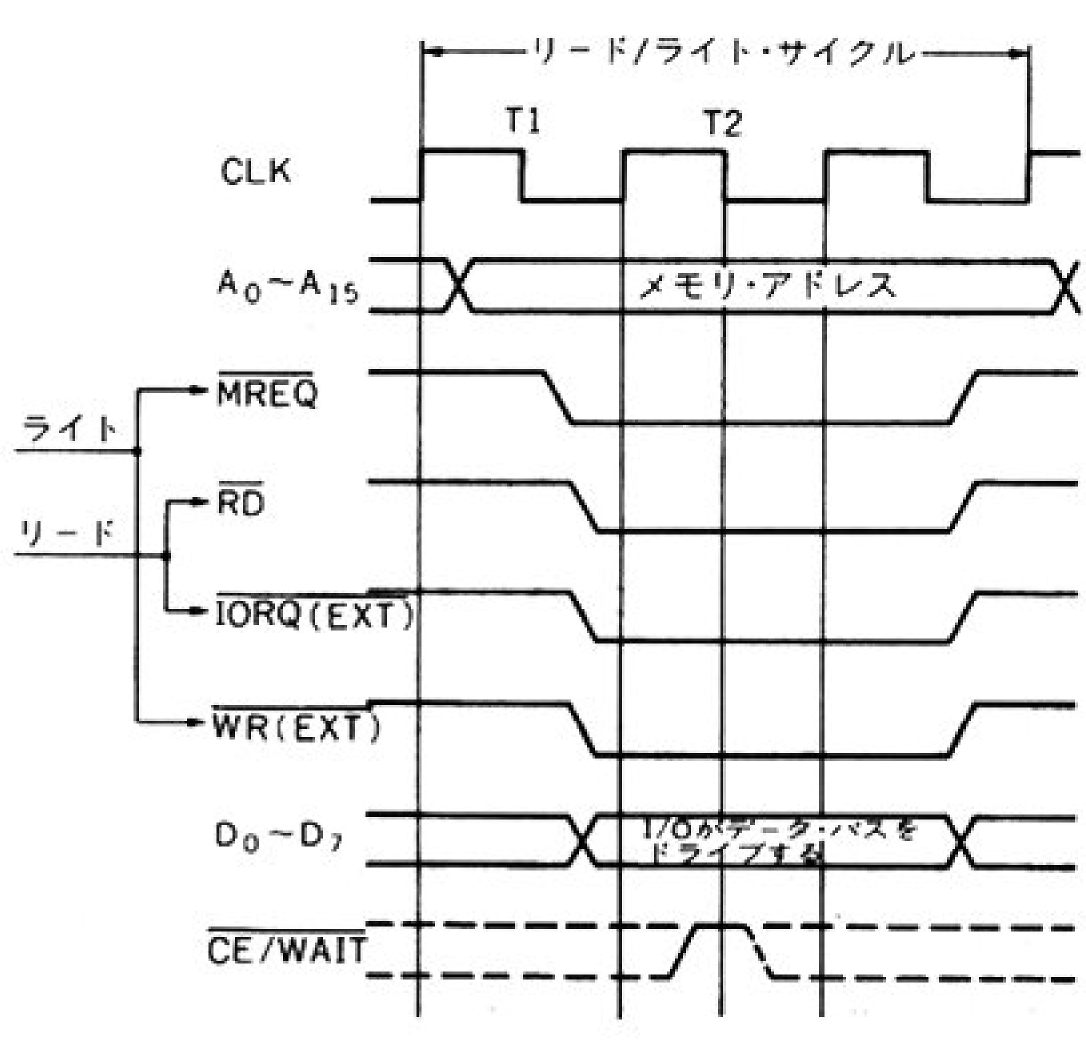
バス・リクエスト・タイミング

図56に Z80 DMA のバス・リクエスト・タイミングを示します．RDY ラインの状態が有効レベルになると，Z80 DMA はクロックの立ち上がり時にこれをサンプリングし，バスが空いている場合（$\overline{\text{BUSREQ}}$ = "H"）

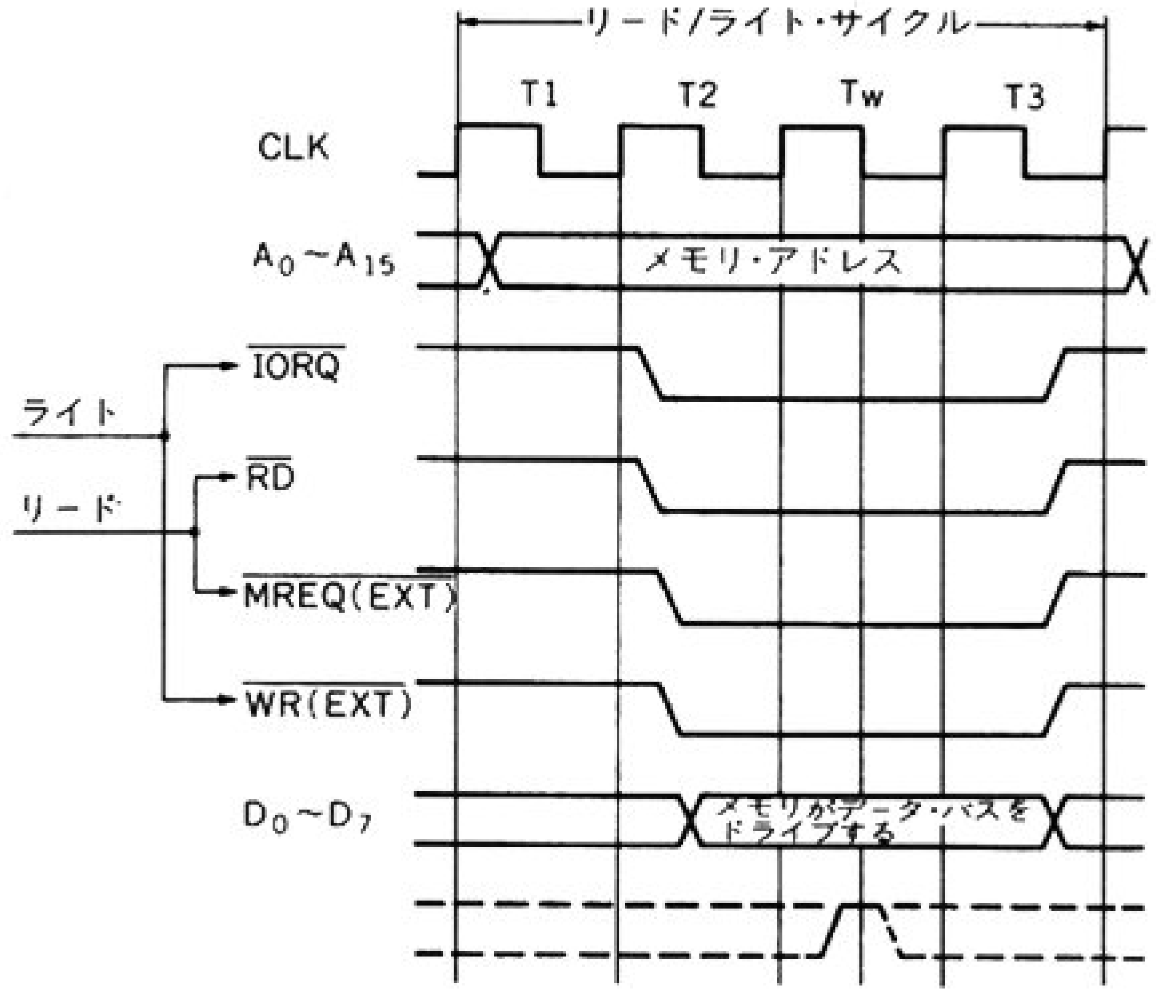
〈図52〉 同時データ転送タイミング（メモリ・サーチ・タイミングを用いた，メモリ → I/O データ転送）



〈図53〉 同時データ転送タイミング（メモリ・サーチ・タイミングを用いた，I/O → メモリ・データ転送）

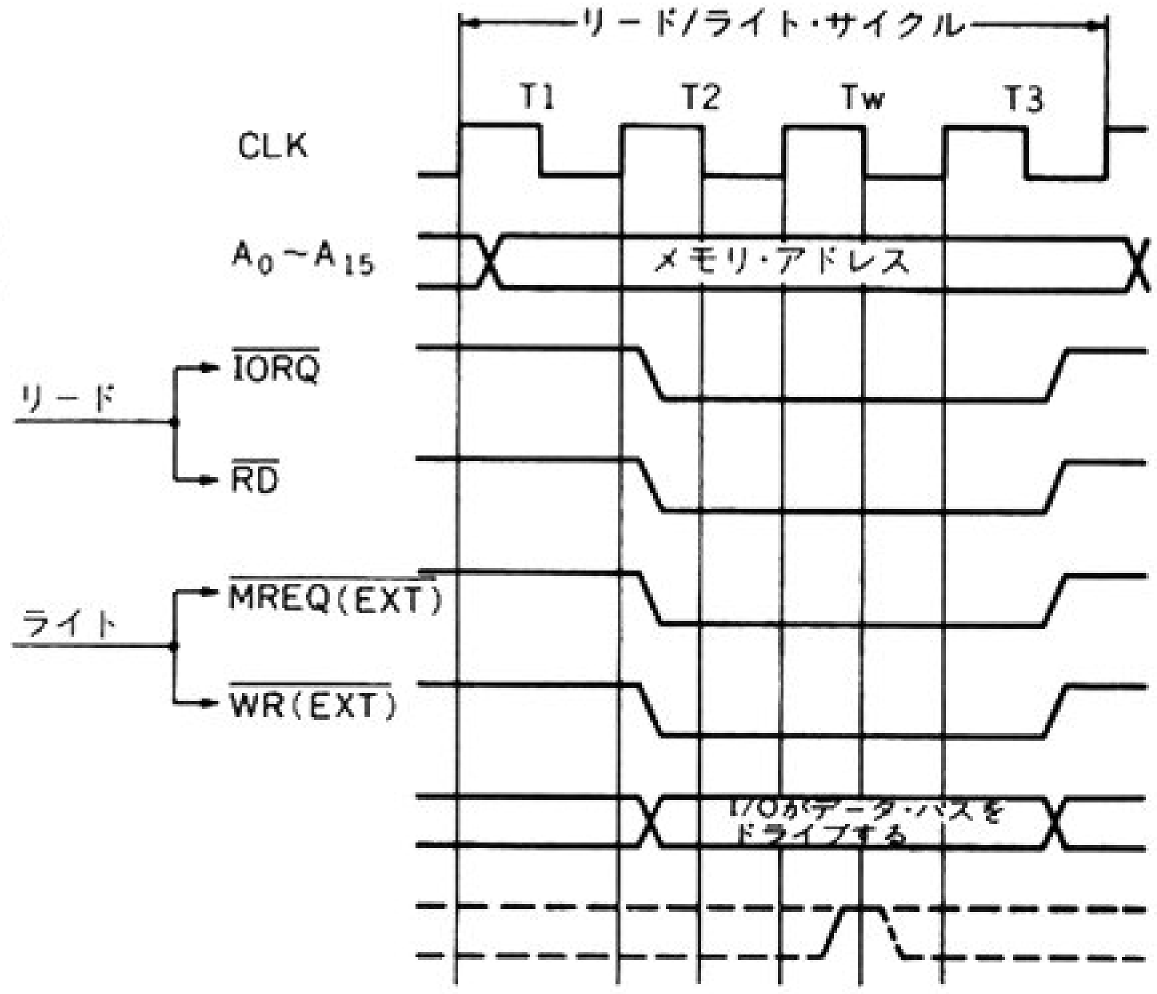


〈図54〉 同時データ転送タイミング(I/Oサーチ・タイミングを用いた，メモリ → I/Oデータ転送)



$A_0 \sim A_{15}$上のアドレスは本来I/Oアドレスであるが，これをメモリ・アドレスとして取り扱う

〈図55〉 同時データ転送タイミング(I/Oサーチ・タイミングを用いた，I/Oメモリ・データ転送)



$A_0 \sim A_{15}$上のアドレスは本来I/Oアドレスであるが，これをメモリ・アドレスとして取り扱う

には，次のクロックの立ち上がり時点に，$\overline{\text{BUSREQ}}$を“L”にして，CPUに対してバスの使用権を要求します．

CPUはその時点に実行しているマシン・サイクルの最後のステートのクロックの立ち上がり時点に，$\overline{\text{BUSREQ}}$をサンプリングし，これが“L”であると次のクロックの立ち上がり時点（次のマシン・サイクルの始め）に，$\overline{\text{BUSAK}}$を“L”にします．

したがって，Z80 DMAがRDYラインが有効レベルになったのを検出してから，CPUがバスの制御権を放棄する（$\overline{\text{BUSAK}}$ = “L”）のに要する時間の最大値は，CPUの1マシン・サイクル（可変である）と1クロック期間の和になります．

Z80 DMAは$\overline{\text{BAI}}$ ($\overline{\text{BUSAK}}$) 入力が2クロック期間，“L”であるのを検出するとDMA動作を開始します．したがってRDYラインが有効レベルになってから，実際にDMAデータ転送が開始されるまでには，最大1マシン・サイクル＋3クロック期間を必要とします．

バス解放タイミング (Bus Release Timing)

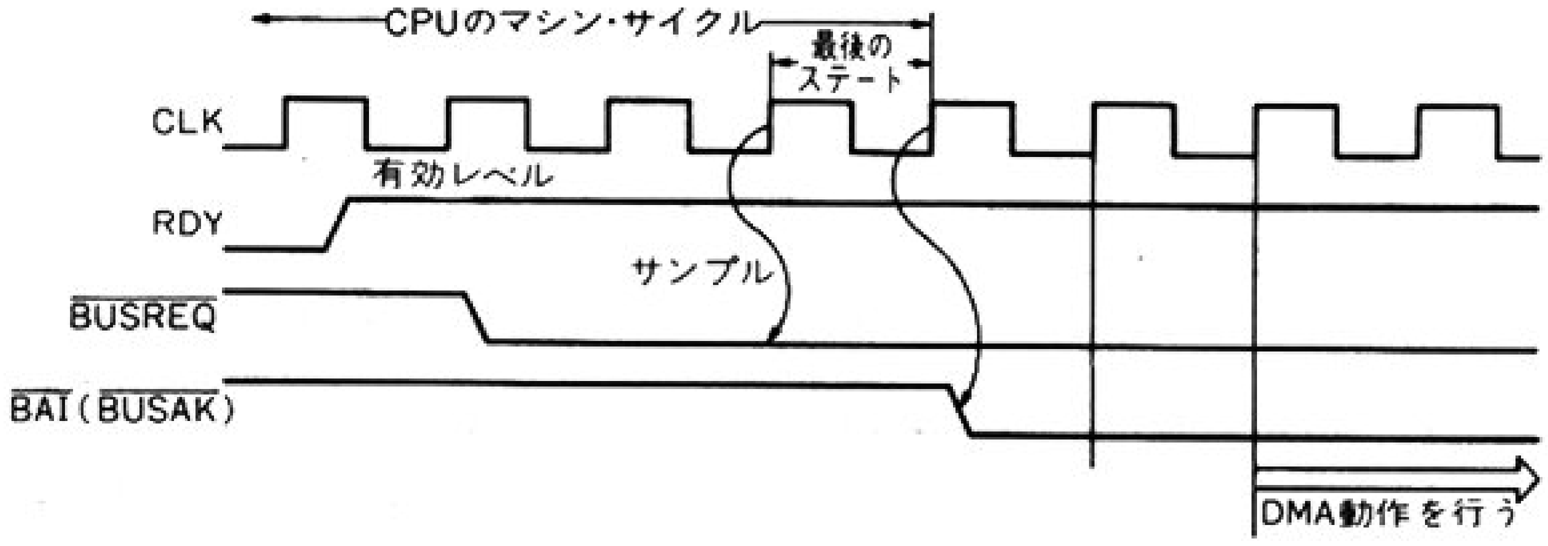
・バイト・モードのときのバス解放

Z80 DMAはバイト・モードのときには，1バイトのDMAデータ転送が終わるごとに，最低1マシン・サイクル期間，バスの制御権をCPUに返します．これがバイト・モードのときのバス解放です．図57にバイト・モードのときの解放タイミングを示します．

バイト・モードでは，Z80 DMAは1データ転送サイクルが終わる（サーチではリード・サイクル，トランスファおよびトランスファ／サーチではライト・サイクルの終わり）直前のクロックの立ち上がり時に，$\overline{\text{BUSREQ}}$を“H”にします．

DMAサイクルが終わる時点よりも1クロックだけ先に$\overline{\text{BUSREQ}}$が“H”になりますが，Z80 CPUは$\overline{\text{BUSREQ}}$が“H”になった時点の1クロック後に動作を再開するので不都合はありません．

バスを解放した後，次のバス・リクエストを行うの



〈図56〉 バス・リクエスト・タイミングRDYの有効レベルはプログラム可であるが，この図では“H”を有効レベルとしている

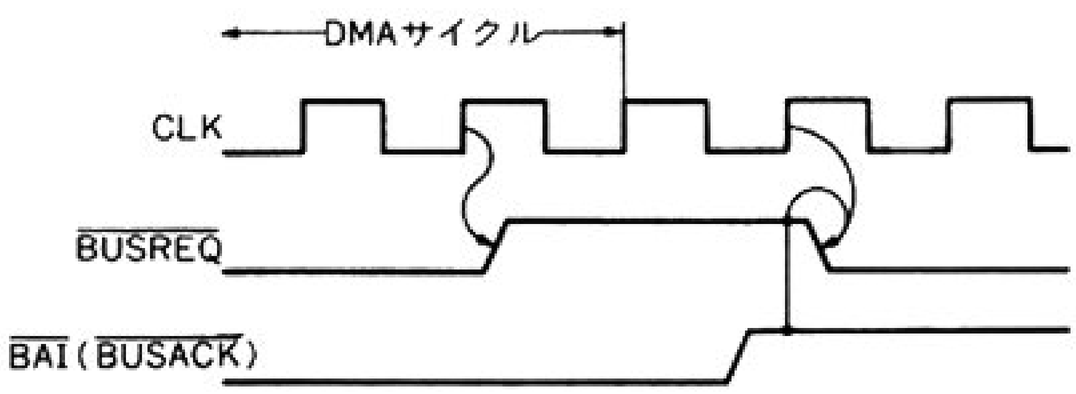
は，$\overline{\rm BUSREQ}$ および $\overline{\rm BAI}$ がともに“H”になった直後のクロックの立ち上がり時点です．もちろん，これには RDY ラインの状態が有効極性にあることが条件です．

■エンド・オブ・ブロック時のバス解放

Z80 DMA は，バースト・モードまたはコンティニュアス・モードで動作しているときには，エンド・オブ・ブロックにより，バス解放を行うことがあります（ストップ・オン・エンド・オブ・ブロックとプログラムされているとき）．

このときには最後のデータの転送終了時のクロックの立ち上がり時点に $\overline{\rm BUSREQ}$ を“H”にします．最後のデータは RDY の状態が有効極性でなくなっていても転送されます．図58にエンド・オブ・ブロック時のバス解放タイミングを示します．

〈図57〉 バイト・モードのバス解放タイミング．DMA サイクルが終わる直前のクロックの立ち上がり時に $\overline{\rm BUSREQ}$ を“H”にする



■マッチ検出時のバス解放

Z80 DMA がバースト・モード，またはコンティニュアス・モードに設定され，かつマッチ時に DMA データ転送を終結するようにプログラミングされているときの，バス解放タイミングを図59に示します．

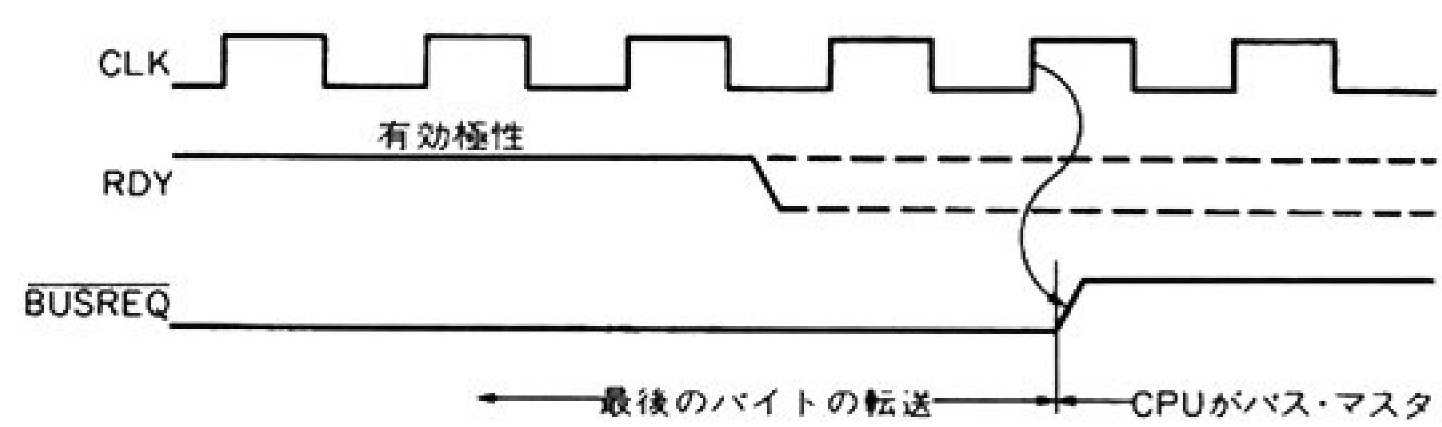
前述のように Z80 DMA の動作はパイプ・ライン化され先読み動作が行われますので，N バイト目のデータがマッチ・バイトと一致するか否かのチェックはN＋1バイト目のデータを転送するのと同時に行います．したがって，N バイト目のデータでマッチが検出されたとしても，データはN＋1バイト転送されます．よってこの場合，Z80 DMA が $\overline{\rm BUSREQ}$ を“H”にするのは，N＋1バイト目のデータの転送が終わったときのクロックの立ち上がり時点になります．

■RDY ラインが有効極性でないときのバス解放

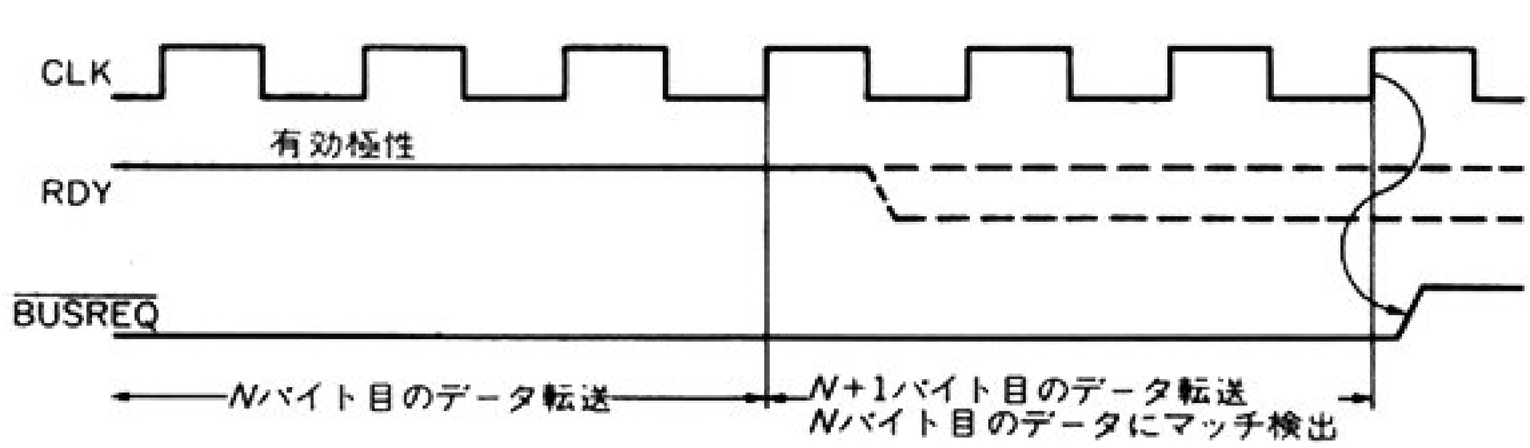
バースト・モードでは，RDY ラインの状能が非有効極性になると，その時点に実行中の DMA サイクルを終了した後のクロックの立ち上がり時点に，$\overline{\rm BUSREQ}$ を“H”にします．図60にバースト・モードにおいて，RDY ラインの状態が非有効極性になったときの，バス解放タイミングを示します．

図61～図63に，RDY ラインの状態が非有効極性となったときの，他の信号の状態をモード別に分けて示します．これらのタイミングは，Z80 DMA に標準タイミングにてメモリ・サーチをやらせた場合を想定し

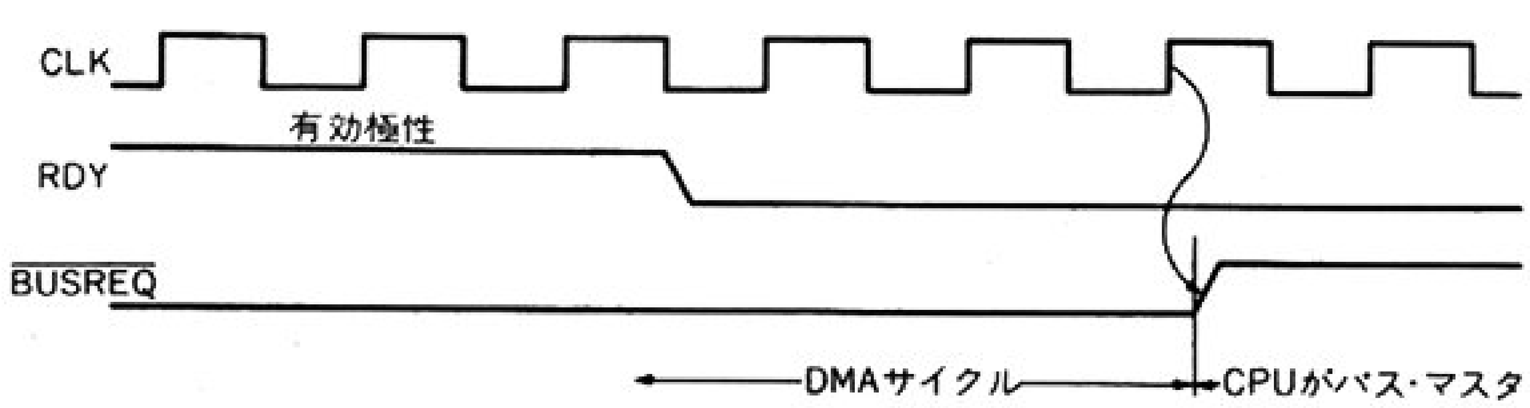
〈図58〉
エンド・オブ・ブロック時のバス解放タイミング
（バースト・モード，またはコンティニュアス・モードのとき）

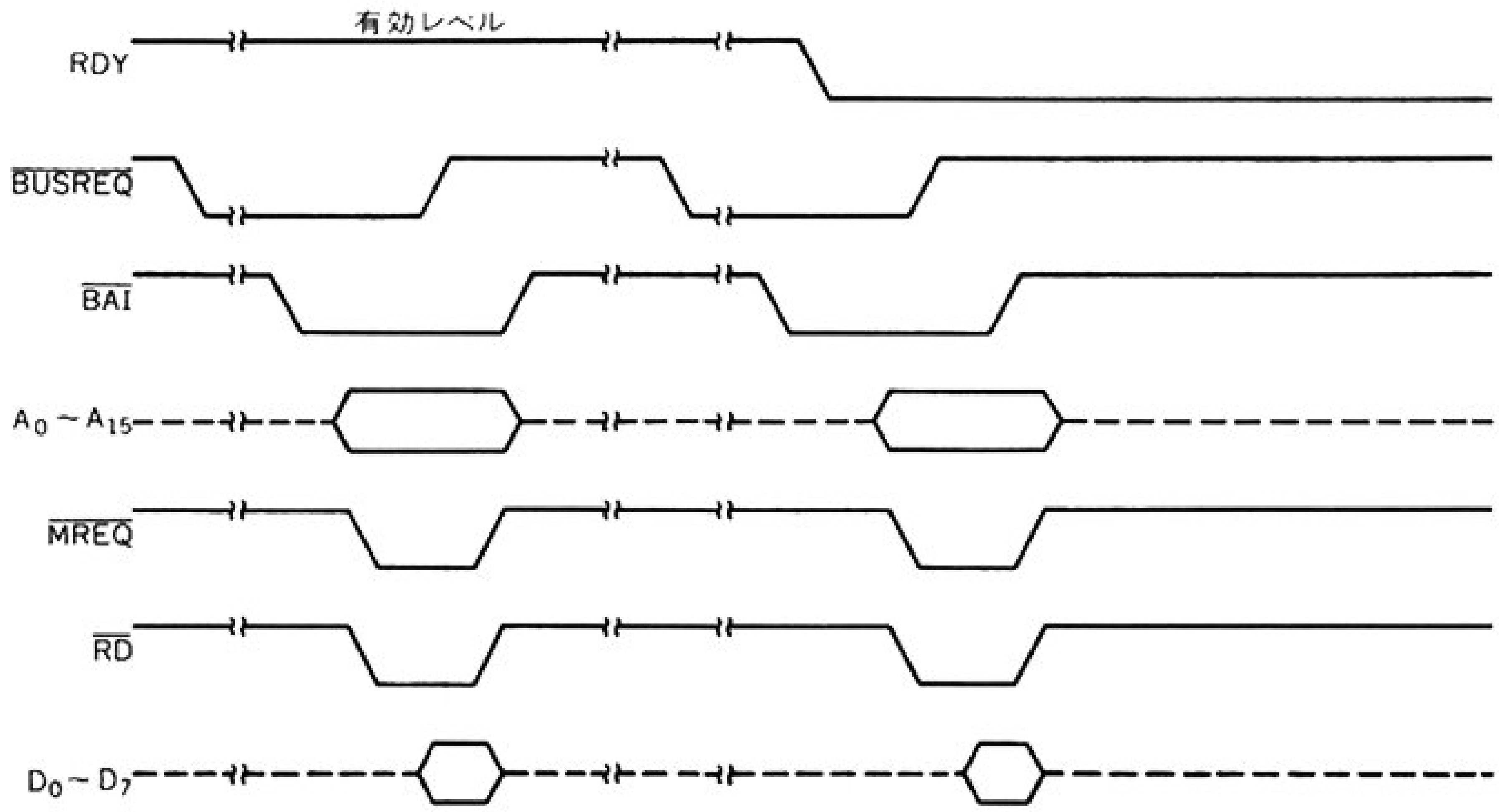


〈図59〉
マッチ検出時のバス解放タイミング（バースト・モード，およびコンティニュアス・モードのとき）

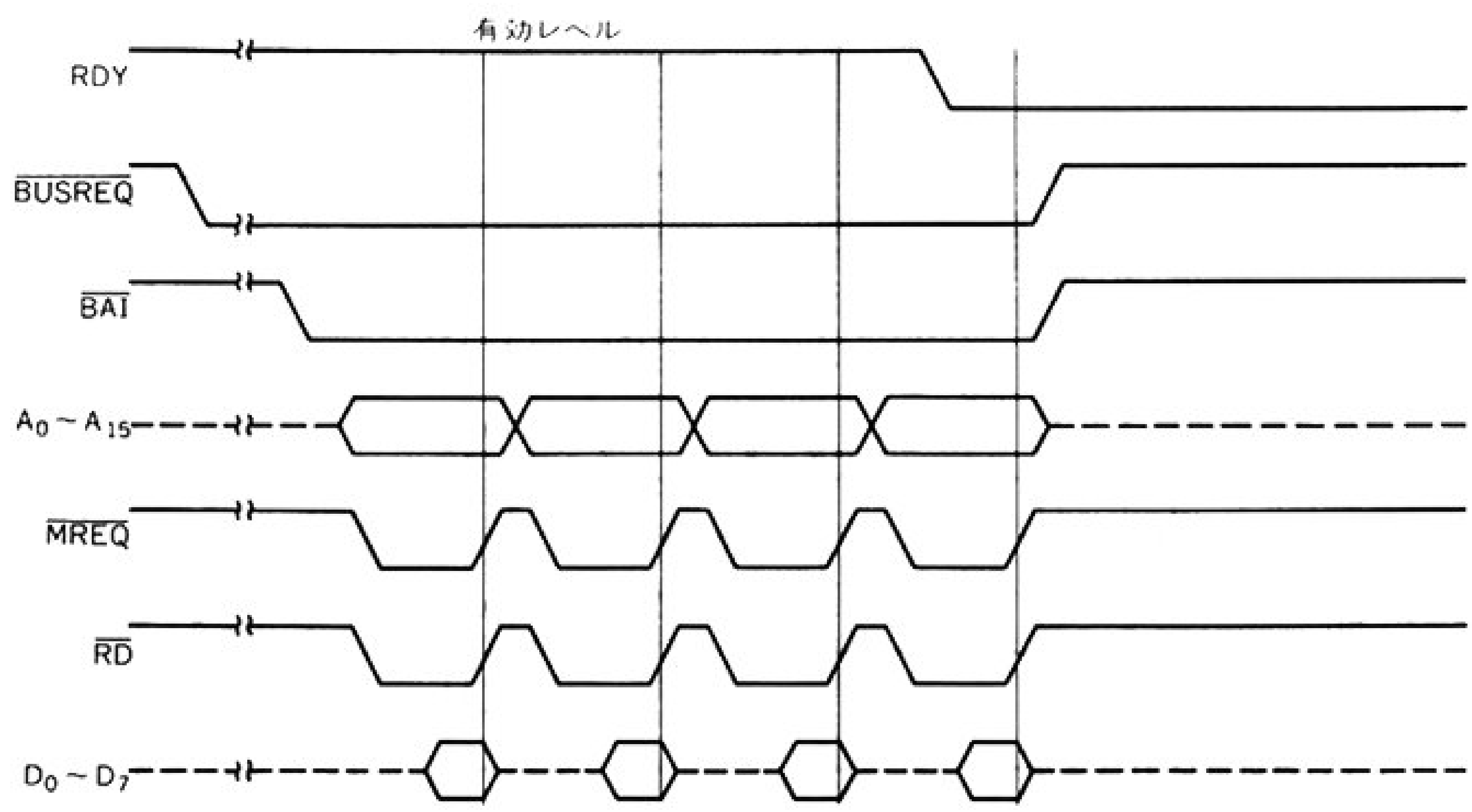


〈図60〉
RDY ラインが非有効極性になったときのバス解放タイミング（バースト・モード）

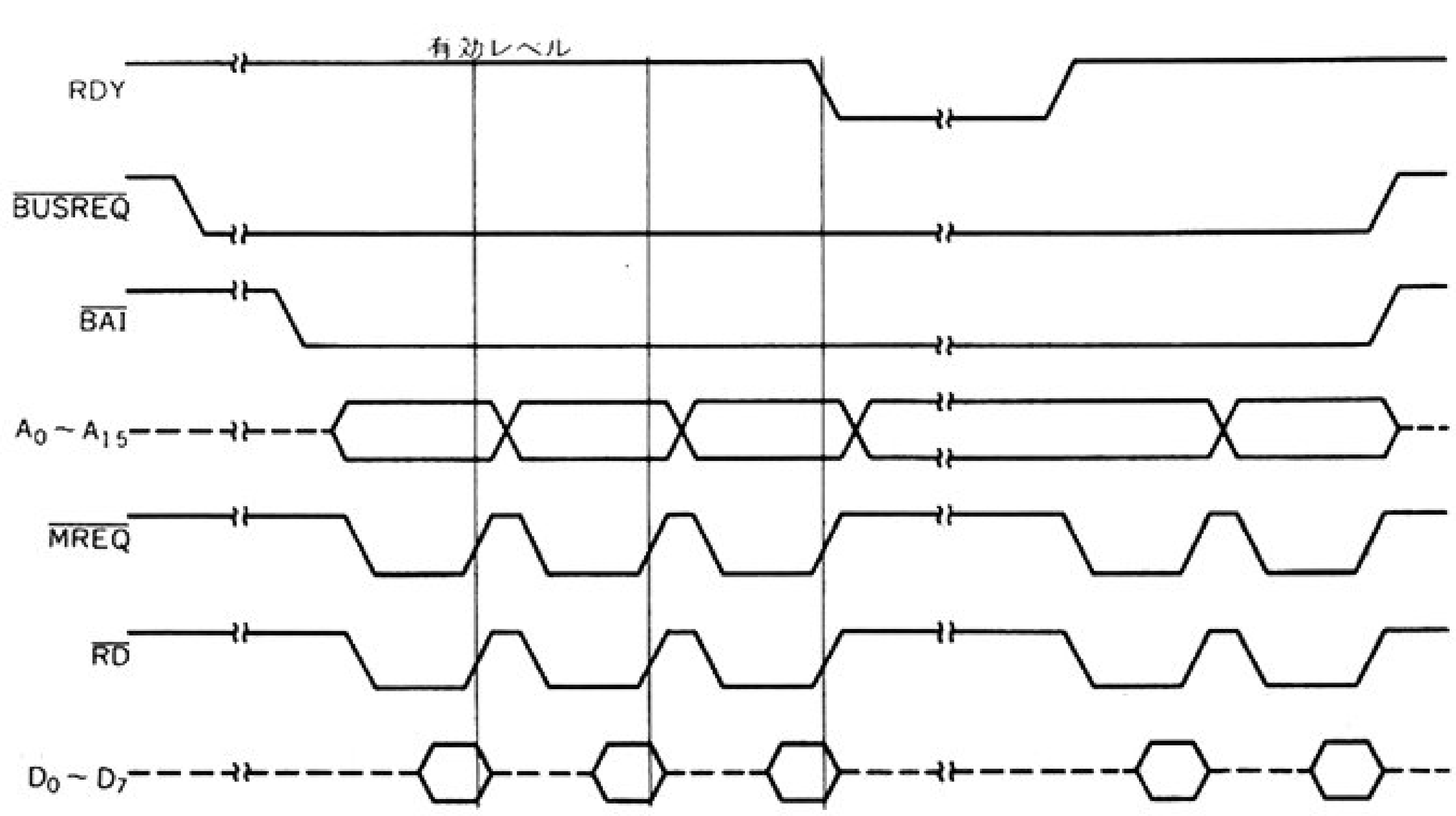




<図61>
RDYラインと他の信号の関係（バイト・モード）



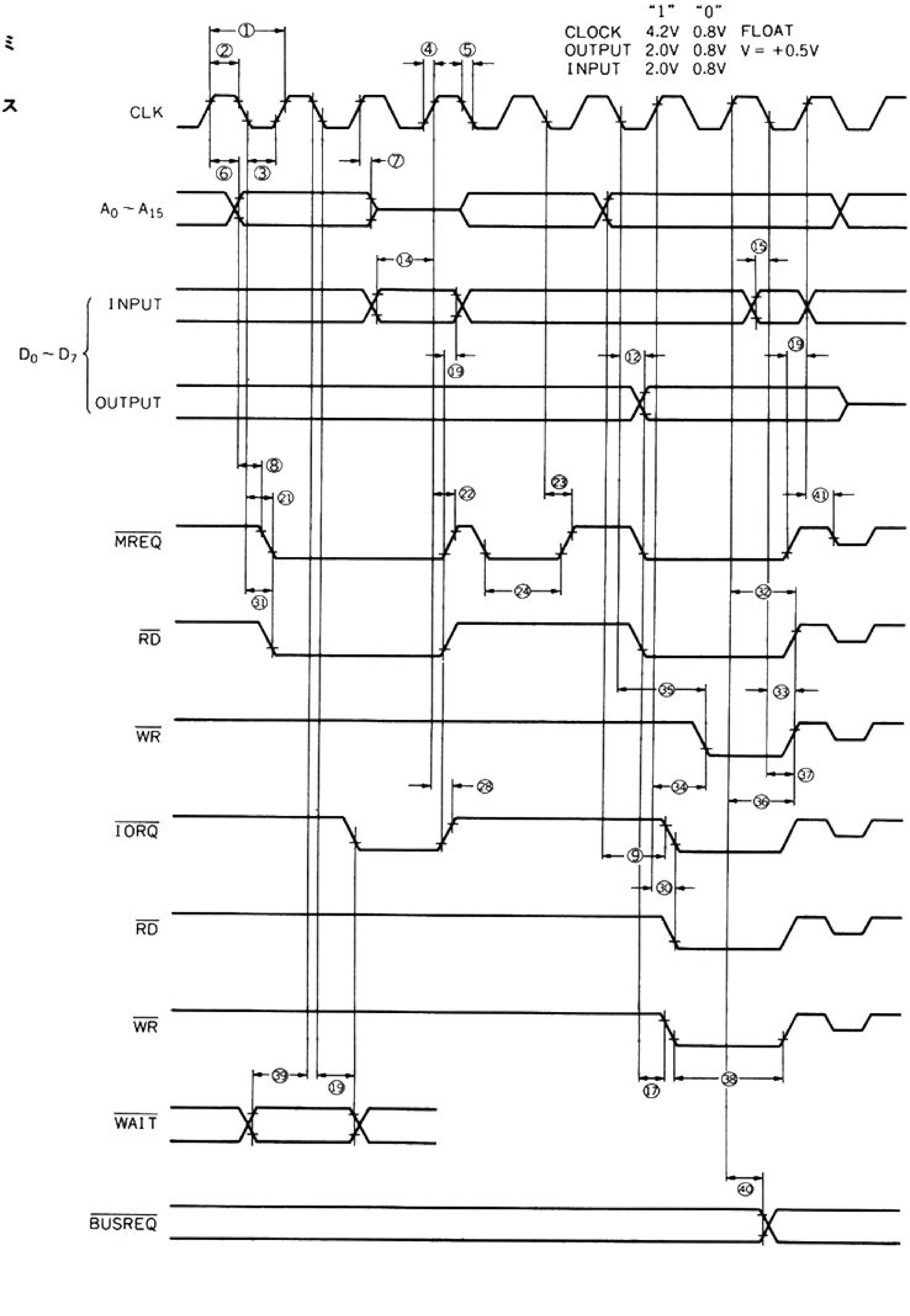
<図62>
RDYラインと他の信号の関係（バースト・モード）



<図63>
RDYラインと他の信号の関係（コンティニュアス・モード）

ています．
　バイト・モード(図61)，およびバースト・モードの場合(図62)には，RDYラインの状態が非有効極性になると，最後のDMAサイクルを終了した後に，$\overline{BUSREQ}$を"H"にし，メモリやI/Oの制御線およびアドレス・バスならびにデータ・バスをハイ・インピーダンスにします．
　コンティニュアス・モードの場合(図63)には，RDYラインの状態が非有効極性になっても，$\overline{BUSREQ}$を"H"にすることはせず，アドレス・バスには次のDMAサイクルで必要なアドレスを乗せたままにします．また，このときメモリやI/Oの制御線を"H"にし，デ

〈図64〉
Z80 DMAのタイミング
(Z80 DMAがバス・マスタ)



ータ・バスはハイ・インピーダンスにします．

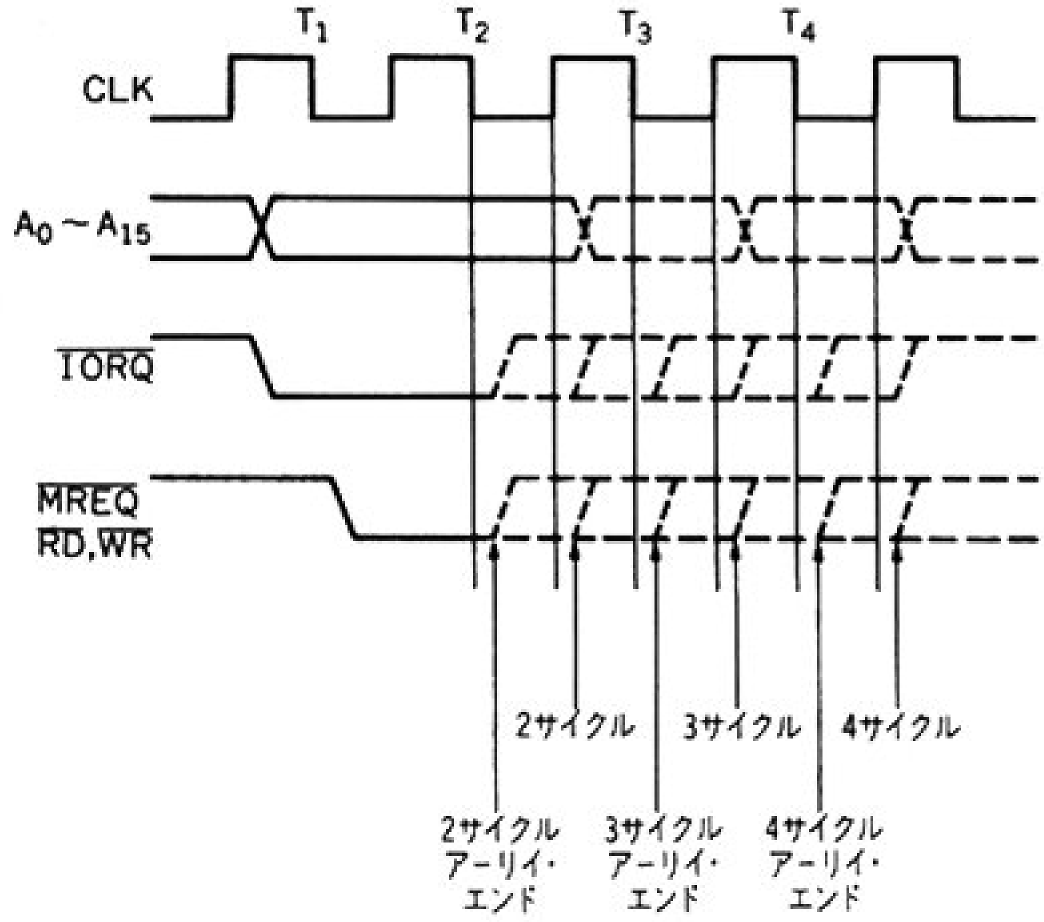
図64にZ80 DMAがバス・マスタであるときのタイミングの詳細を示します．

可変サイクル・タイミング

Z80 DMAには可変タイミングという機能があり，DMAデータ転送時のサイクル長を2～4クロック・サイクルの範囲で変更することができます．また，サイクル長だけでなく，メモリおよびI/Oの制御信号の立ち上がりタイミングを2分の1クロックだけ前に進めることができます(アーリイ・エンド)．可変タイミングに関するプログラミングは，WR1およびWR2によ

番号	記号	パラメータ	Z80 DMA Min(ns)	Z80 DMA Max(ns)	Z80A DMA Min(ns)	Z80A DMA Max(ns)
1	TcC	Clock Cycle Time	400		250	
2	TwCh	Clock Width (High)	180	2000	110	2000
3	TwCl	Clock Width (Low)	180	2000	110	2000
4	TrC	Clock Rise Time		30		30
5	TfC	Clock Fall Time		30		30
6	TdA	Address Output Delay		145		110
7	TdC(Az)	Clock ↑ to Address Float Delay		110		90
8	TsA(MREQ)	Address to $\overline{MREQ}$ ↓ Setup (Memory Cycle)	(2)+(5)−75		(2)+(5)−75	
9	TsA(IRW)	Address Stable to $\overline{IORQ}$, $\overline{RD}$, $\overline{WR}$ ↓ Setup (I/O Cycle)	(1)−80		(1)−70	
10	TdRW(A)	$\overline{RD}$, $\overline{WR}$ ↑ to Addr. Stable Delay	(3)+(4)−40		(3)+(4)−50	
11	TdRW(Az)	$\overline{RD}$, $\overline{WR}$ ↑ to Addr. Float	(3)+(4)−60		(3)+(4)−45	
12	TdCf(DO)	Clock ↓ to Data Out Delay		230		150
13	TdCr(Dz)	Clock ↑ to Float Delay (Write Cycle)		90		90
14	TsDI(Cr)	Data In to Clock ↑ Setup (Read cycle when rising edge ends read)	50		35	
15	TsDI(Cf)	Data In to Clock ↓ Setup (Read cycle when falling edge ends read)	60		50	
16	TsDO(WfM)	Data Out to $\overline{WR}$ ↓ Setup (Memory Cycle)	(1)−210		(1)−170	
17	TsDO(WfI)	Data Out to $\overline{WR}$ ↓ Setup (I/O cycle)	100		100	
18	TdWr(DO)	$\overline{WR}$ ↑ to Data Out Delay	(3)+(4)−80		(3)+(4)−70	
19	Th	Hold Time for Any Specified Setup Time	0		0	
20	TdCf(Mf)	Clock ↓ to $\overline{MREQ}$ ↓ Delay		100		85
21	TdCr(Mr)	Clock ↑ to $\overline{MREQ}$ ↑ Delay		100		85
22	TdCf(Mr)	Clock ↓ to $\overline{MREQ}$ ↑ Delay		100		85
23	TwMl	$\overline{MREQ}$ Low Pulse Width	(1)−40		(1)−30	
24	TwMh	$\overline{MREQ}$ High Pulse Width	(2)+(5)−30		(2)+(5)−20	
25	TdCr(If)	Clock ↑ to $\overline{IORQ}$ ↓ Delay		90		75
26	TdCr(Ir)	Clock ↑ to $\overline{IORQ}$ ↑ Delay		100		85
27	TdCf(Ir)	Clock ↓ to $\overline{IORQ}$ ↑ Delay		110		85
28	TdCr(Rf)	Clock ↑ to $\overline{RD}$ ↓ Delay		100		85
29	TdCf(Rf)	Clock ↓ to $\overline{RD}$ ↓ Delay		130		95
30	TdCr(Rr)	Clock ↑ to $\overline{RD}$ ↑ Delay		100		85
31	TdCf(Rr)	Clock ↓ to $\overline{RD}$ ↑ Delay		110		85
32	TdCr(Wf)	Clock ↑ to $\overline{WR}$ ↓ Delay		80		65
33	TdCf(Wf)	Clock ↓ to $\overline{WR}$ ↓ Delay		90		80
34	TdCr(Wr)	Clock ↑ to $\overline{WR}$ ↑ Delay		100		80
35	TdCf(Wr)	Clock ↓ to $\overline{WR}$ ↑ Delay		100		80
36	TwWl	$\overline{WR}$ Low Pulse Width	(1)−40		(1)−30	
37	TsWA(Cf)	$\overline{WAIT}$ to Clock ↓ Setup	70		70	
38	TdCr(B)	Clock ↑ to $\overline{BUSREQ}$ Delay		150		100
39	TdCr(Iz)	Clock ↑ to $\overline{IORQ}$, $\overline{MREQ}$, $\overline{RD}$, $\overline{WR}$ Float Delay		100		80

〈図65〉 可変サイクル・タイミング



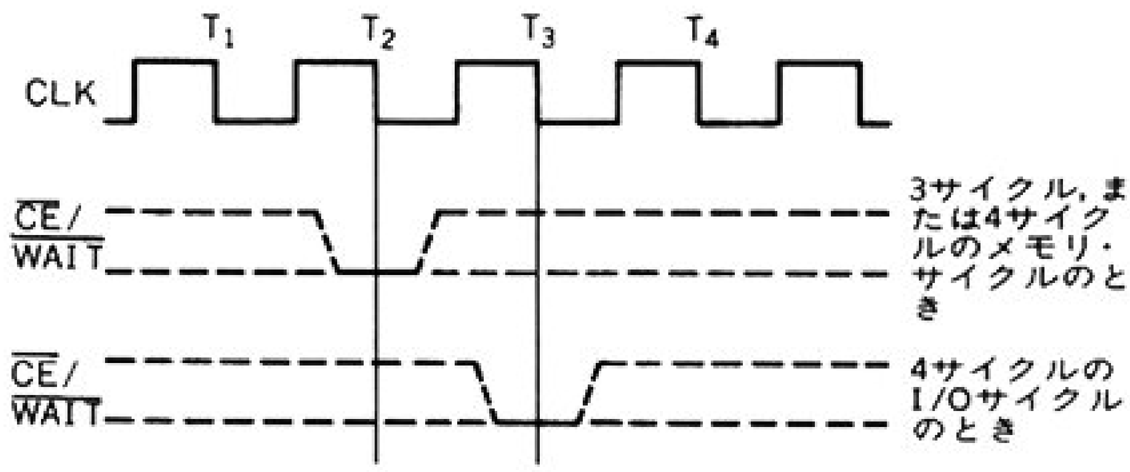
り行います．

図65に可変サイクル・タイミングにおける各制御信号（$\overline{MREQ}$，$\overline{IORQ}$，$\overline{RD}$，$\overline{WR}$）とクロック，それにアドレス・バスの関係を示します．

この図からわかるように，可変サイクル・タイミングでは，$\overline{IORQ}$ は T1 の始めから出力されます．Z80標準タイミングでは，$\overline{IORQ}$ は T2 の始めに出力されます．また $\overline{WR}$ のタイミングも標準タイミングとは異なります．標準タイミングでは，メモリ・ライト・サイクルにおける $\overline{WR}$ は T2 の中頃に出力されますが，この可変サイクル・タイミングでは常に T1 の中頃に出力されます．

また可変サイクル・タイミングを用いたときには，メモリ・サイクルでは３サイクル，または４サイクル・タイミング，I/O サイクルでは４サイクル・タイミングのときにしか，ウェイト機能が働かないので注意を要します．Z80 DMA はメモリ・サイクルでは T2 に

〈図66〉 可変サイクル・タイミングにおける，ウェイト信号（$\overline{CE/WAIT}$）のサンプリング・タイミング



て，I/O サイクルでは T3 において $\overline{CE/WAIT}$ 信号をサンプリングします（図66）．

パルス発生タイミング

Z80 DMA ではプログラミングにより，256 バイトのデータが DMA 転送されるごとに，$\overline{INT}$ ライン上にパルスを発生する機能を持っています．このパルスの幅は１DMA 転送サイクル期間です．したがって，Z80 標準タイミングでメモリ ➡ I/O のデータ転送を行っているときには７クロック期間となり，メモリ・サーチを行っているときには３クロック期間となります．

図67にトランスファおよびメモリ・サーチ時のパルス・タイミングを示します（標準タイミング）．

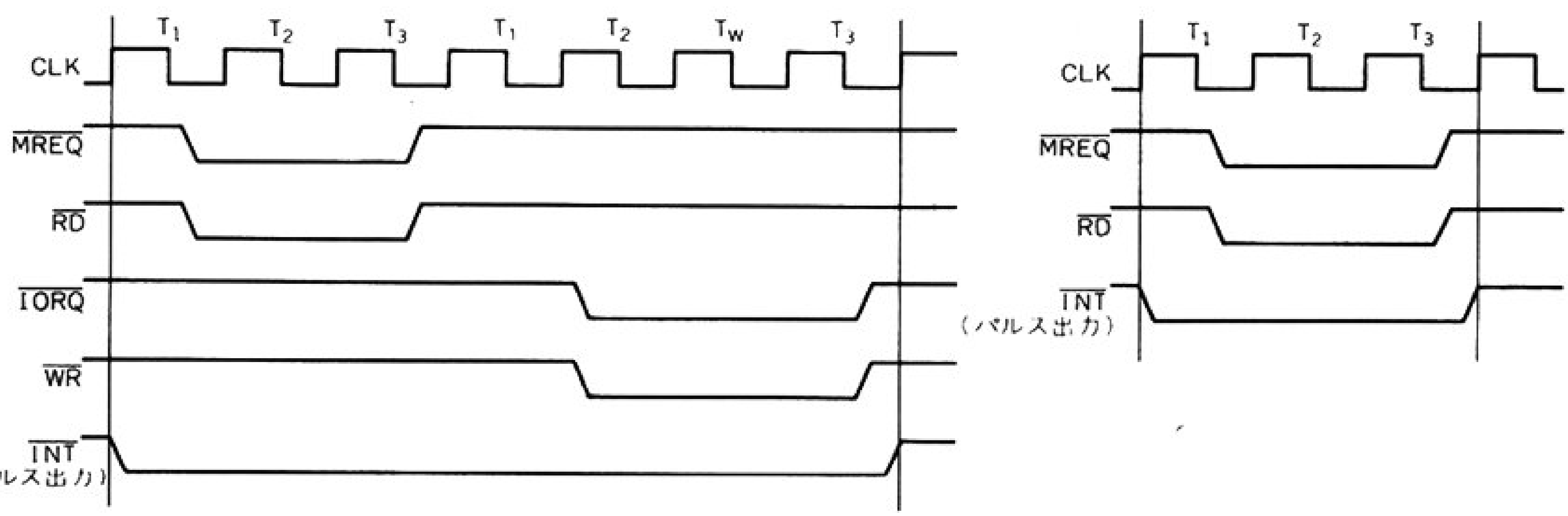
Z80 DMAの使いかた（ハードウェア）

ここでは Z80 DMA をハードウェアとして動作させるために必要なことがらを述べます．

Z80 CPU とのインターフェース

Z80 DMA を使う上で，もっともたいせつなことは

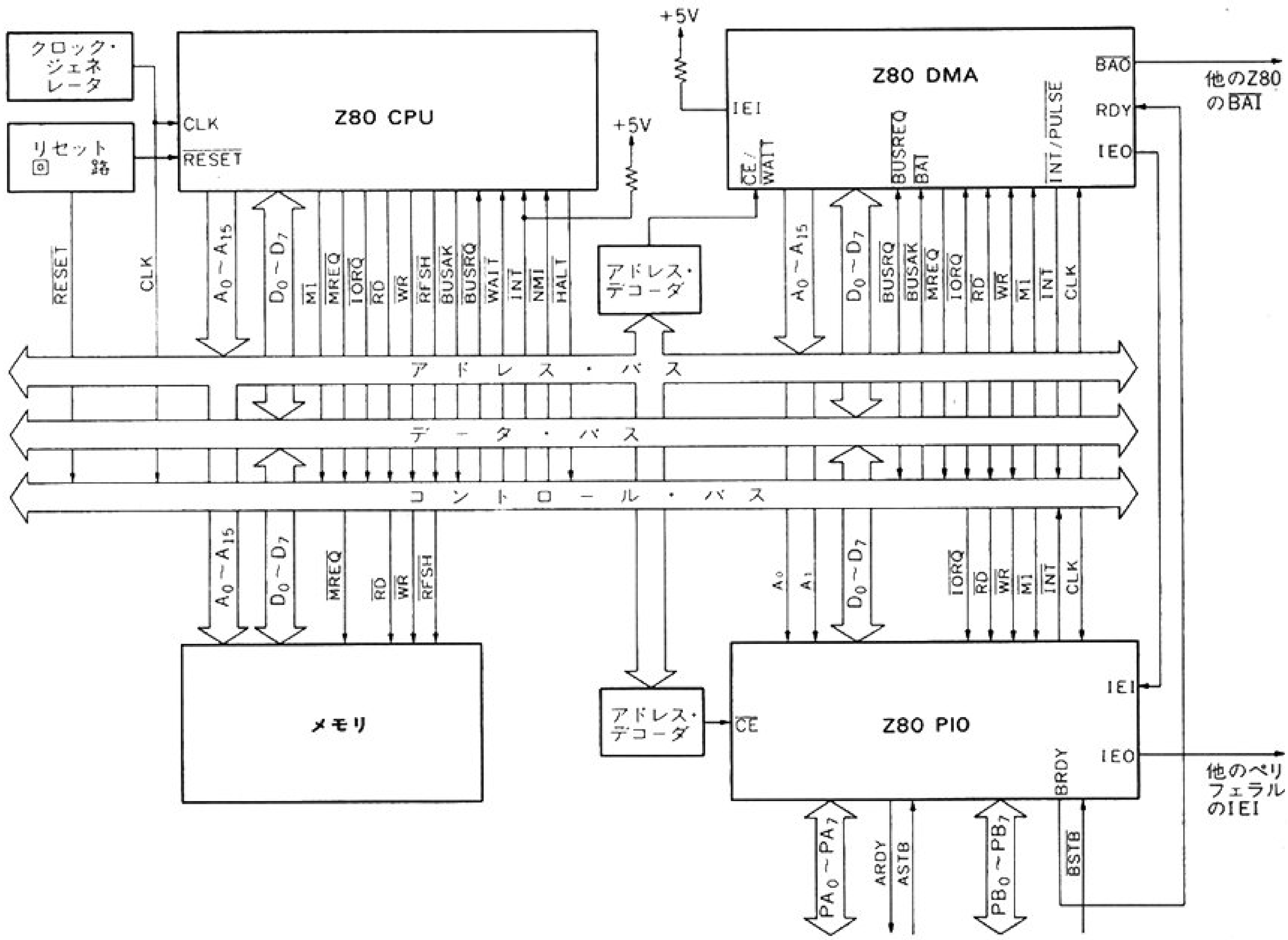
〈図67〉 パルス出力（$\overline{INT}$）のタイミング



（a） トランスファ時のパルス出力（標準タイミング）

（b） メモリ・サーチ時のパルス出力（標準タイミング）

〈図68〉 Z80 CPU，-DMA，-PIO，およびメモリとのインターフェース



Z80 CPU とのインターフェースであることはいうまでもありません．図68に Z80 CPU と Z80 DMA，それにメモリと Z80 PIO とのインターフェースを示します．

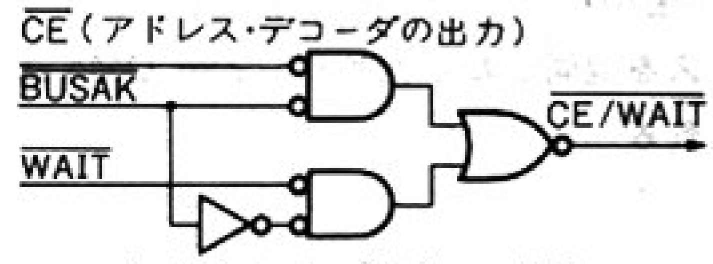
Z80 CPU と Z80 ファミリのペリフェラルとのインターフェースは，Z80 CPU の信号線のほとんどをそのまま接続することにより行われます．このことは Z80 DMA についてもいうことができます．IEI/IEO の接続は，Z80 ファミリ LSI の割り込みデイジー・チェインの約束ごとにしたがって行われます．この図では割り込み優先順位は，Z80 DMA，Z80 PIO の順になっています．

CPU の出力，$\overline{\text{BUSAK}}$ は Z80 DMA の $\overline{\text{BAI}}$ に接続され，$\overline{\text{BAO}}$ は DMA が複数個ある場合，次にバス使用優先順位の高い DMA の $\overline{\text{BAI}}$ に接続されます．

$\overline{\text{CE}}/\overline{\text{WAIT}}$ マルチプレクサ

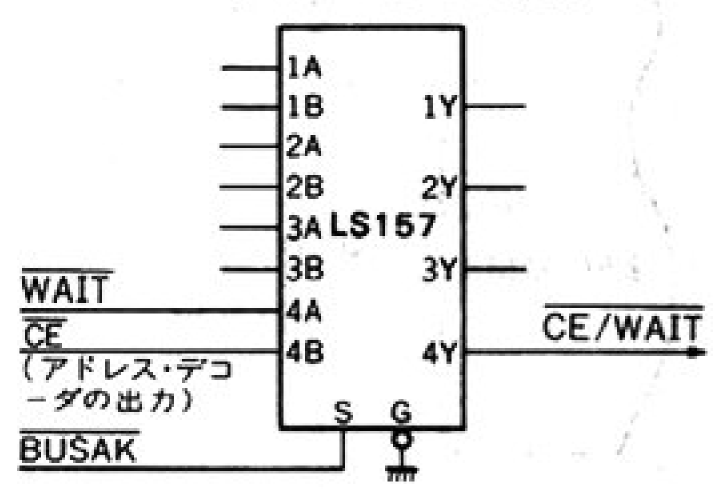
WR5 のプログラミングにより，Z80 DMA の $\overline{\text{CE}}/\overline{\text{WAIT}}$ ピンは，$\overline{\text{CE}}$（チップ・イネーブル）および $\overline{\text{WAIT}}$ 双方の働きをすることができます．この場合，$\overline{\text{BUSAK}}$ が"H"であるときには（CPU がバス・マスタ），$\overline{\text{CE}}$ として働き，$\overline{\text{BUSAK}}$ が"L"であるときには（DMA がバス・マスタ），$\overline{\text{WAIT}}$ として働きます．

したがって，$\overline{\text{CE}}/\overline{\text{WAIT}}$ ピンに $\overline{\text{CE}}$ および $\overline{\text{WAIT}}$ 両方の働きをさせたいときには，$\overline{\text{CE}}$ および $\overline{\text{WAIT}}$ 信号を，$\overline{\text{BUSAK}}$ により切り替える（マルチプレクスする）必要があります．これを行うロジックを $\overline{\text{CE}}/\overline{\text{WAIT}}$ マルチプレクサと呼びます．図69に $\overline{\text{CE}}/\overline{\text{WAIT}}$



（a） ゲートで構成した場合（LS02 1個でできる）

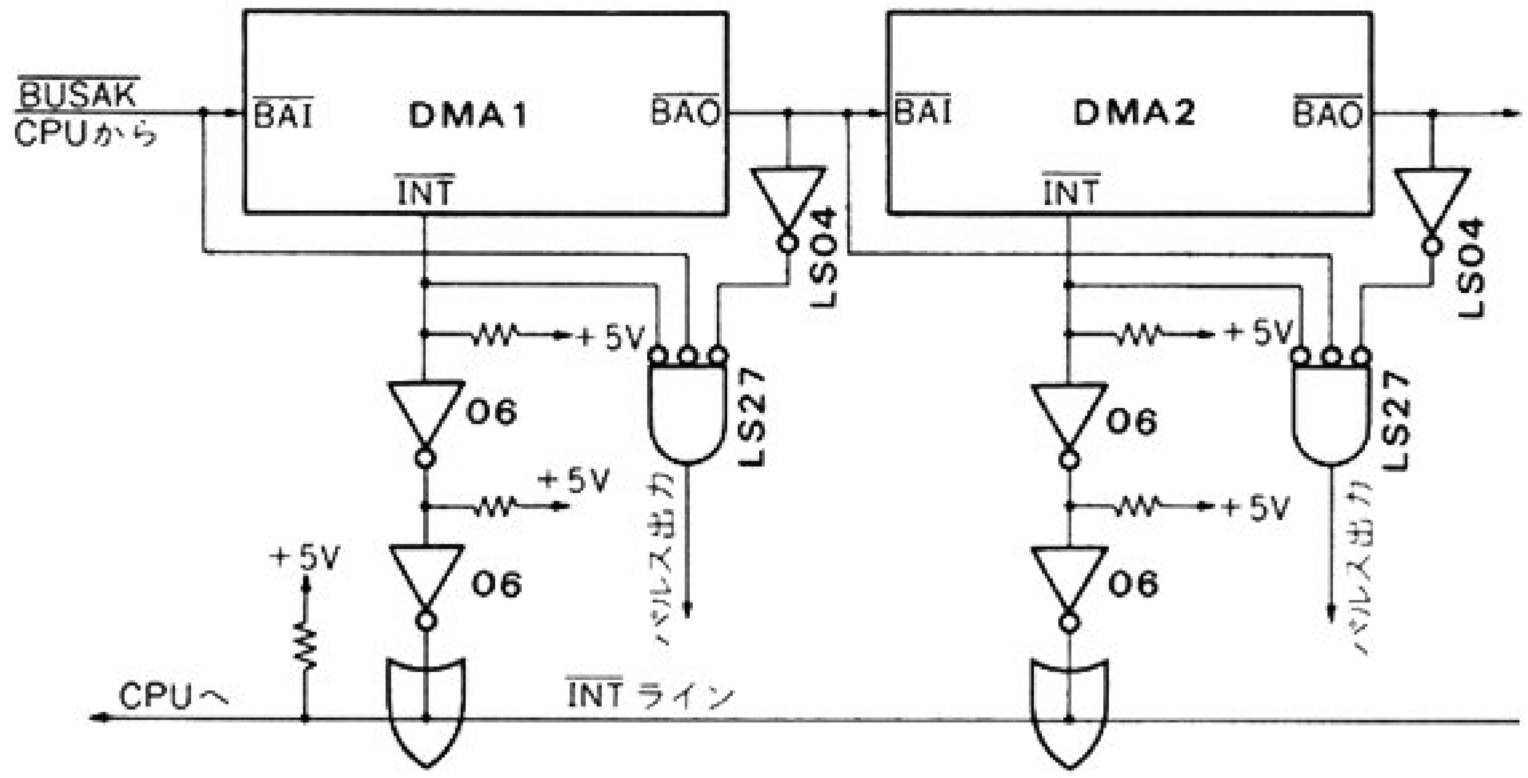
〈図69〉
$\overline{\text{CE}}/\overline{\text{WAIT}}$ マルチプレクサ



（b） マルチプレクサを用いた場合

マルチプレクサの例を示します．この図にはゲートで構成した場合と，マルチプレクサ（LS157）を用いた場合の両方を示してあります．

〈図71〉 2個のZ80 DMAからパルスを抽出する回路



パルス抽出回路

WR4のプログラミングにより，256バイトのDMAデータ転送を行うごとに，$\overline{\text{INT}}$ライン上にパルスを出力させることができます．Z80 DMAはこのパルスを$\overline{\text{BUSAK}}$が"L"であるときに，$\overline{\text{INT}}$ライン上に出力します．また，Z80 DMAは$\overline{\text{BUSAK}}$が"L"であるときには割り込み要求を出すことはしません．

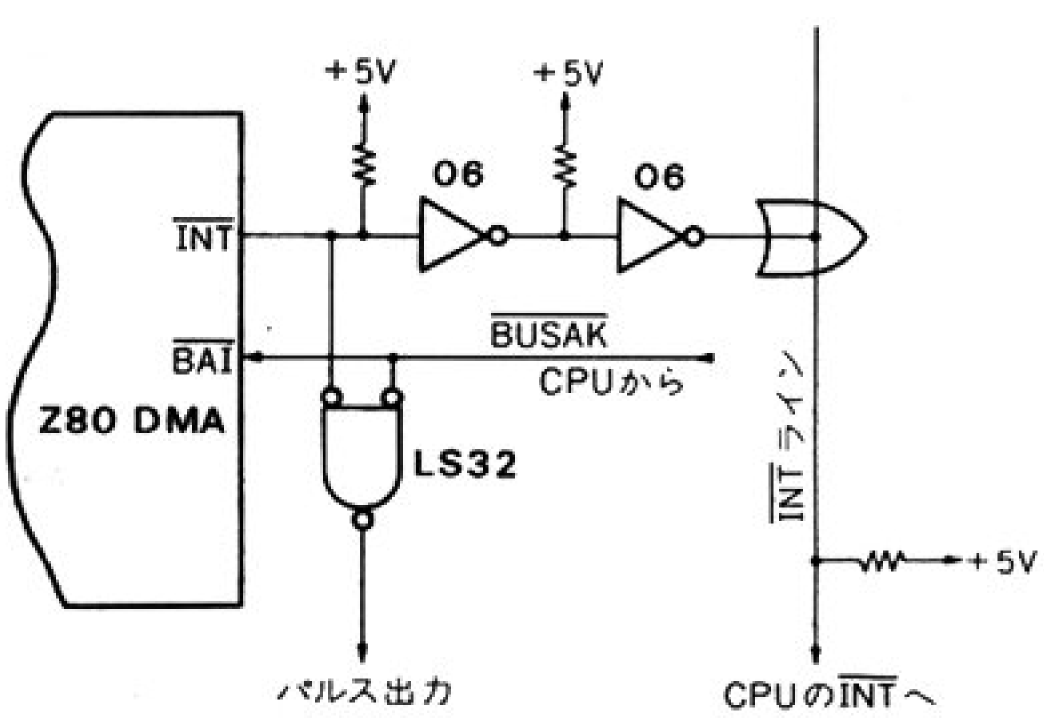
したがって，ことZ80 DMAに関する限り，$\overline{\text{BUSAK}}$と$\overline{\text{INT}}$をAND結合することにより，パルスを抽出することができます．

ところが$\overline{\text{INT}}$ライン上に他のLSI（たとえばPIO）がぶらさがっているような場合には，誤ったパルスを取り出してしまう可能性があります．というのはZ80 PIOには$\overline{\text{BUSAK}}$ピンなどありませんので，PIOは$\overline{\text{BUSAK}}$状態であろうとなかろうと，おかまいなしに割り込み要求を出してくることだってありうるからです．

これを避けるには，パルスを抽出したいZ80 DMAの$\overline{\text{INT}}$ラインを他のLSIの$\overline{\text{INT}}$と直接的にワイヤードORするのをやめ，バッファを介して接続することにより，他のLSIの$\overline{\text{INT}}$を論理的に切り離す必要があります．

図70にパルス抽出回路の例を示します．この例では，$\overline{\text{INT}}$を切り離すのに7406のバッファを2段設けています．図71に2個のZ80 DMAを用い，それぞれからパルスを抽出する場合のロジックを示します．この場合には DMAがバス・マスタであることを保証するために，各DMAの$\overline{\text{BAI}}$が"L"で$\overline{\text{BAO}}$が"H"であることを条件にして，$\overline{\text{INT}}$からパルスを取り出しています．

〈図70〉 パルス抽出回路．他のLSIの割り込み要求とパルスを混同しないために，Z80 DMAの$\overline{\text{INT}}$にバッファを入れる



Z80 DMAは自分がバス・マスタになると$\overline{\text{BAO}}$を"H"にして，バス優先順位の低いDMAがバス・マスタになることを禁止するからです．なお，図70の例ではパルス出力は"L"有効ですが，図71の場合には"H"有効です．

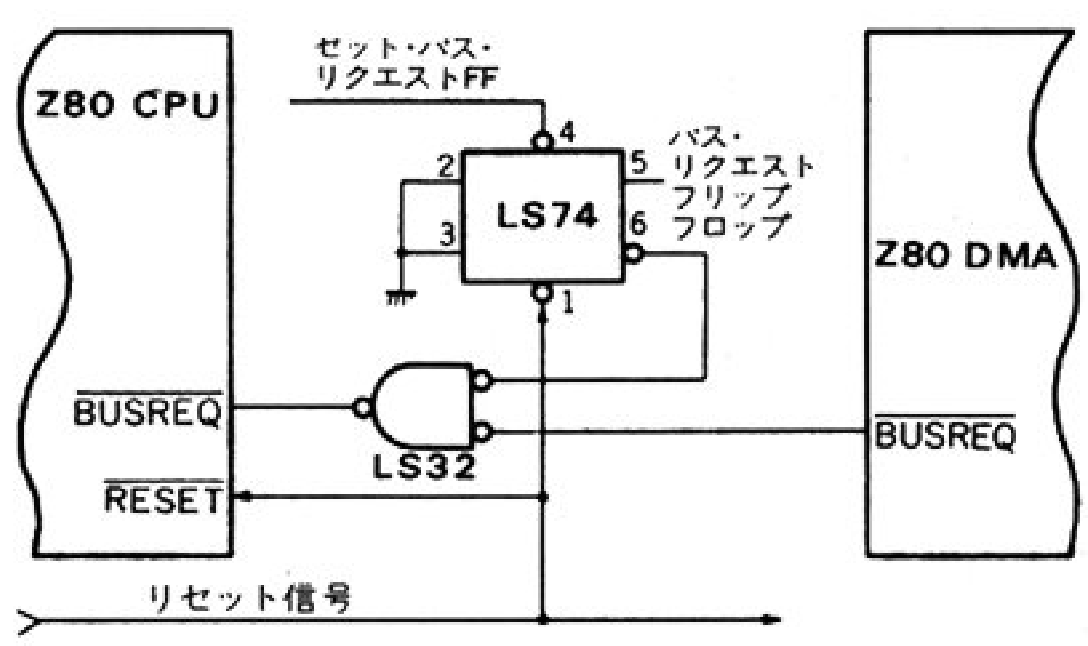
ハング・アップ防止回路

Z80 DMAにはリセット入力がありませんので，Z80 DMAが$\overline{\text{BUSREQ}}$を出しっぱなしの状態になったときには，システムはハング・アップしてしまいます．

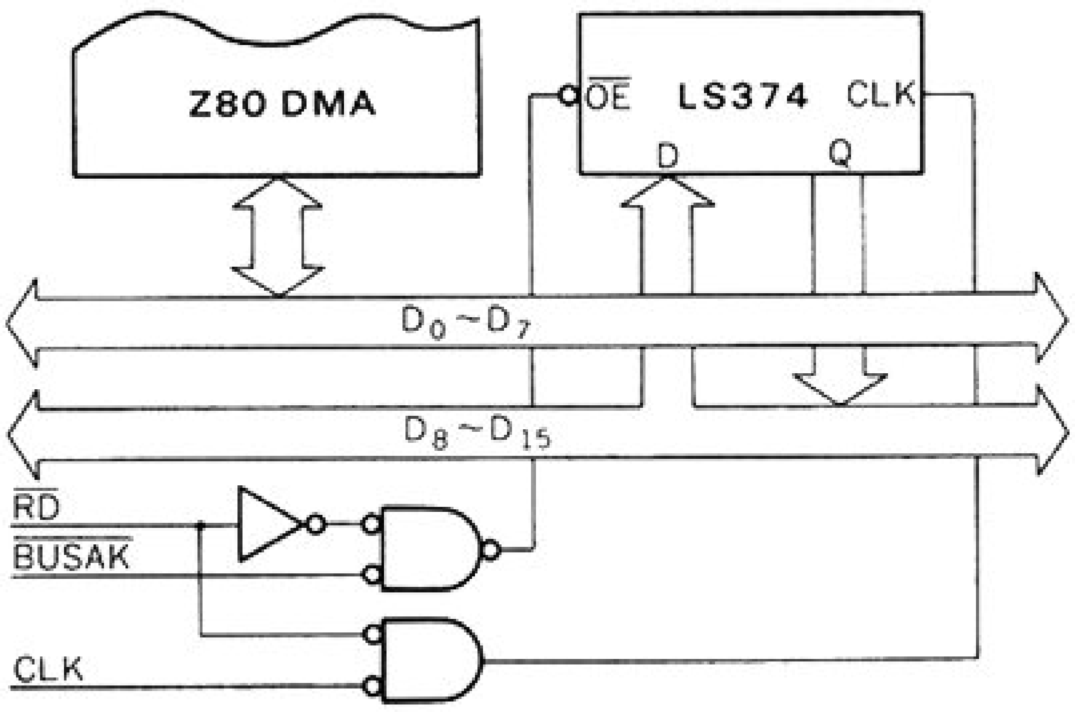
このような状態は，Z80 DMAをコンティニュアス・モードで，かつオート・リスタートと設定し，フォース・レディによりDMA動作を開始することにより，簡単に作ることができます．

この場合にはZ80 DMAは永久に$\overline{\text{BUSRQ}}$を出し続けますのでCPUは動くことができません．たとえシステム・リセット・スイッチを押してリセット信号を出したとしても，Z80 DMAにはリセット入力がありませんから，リセット信号などおかまいなしに$\overline{\text{BUSRQ}}$

〈図72〉 ハング・アップ防止回路



〈図73〉 16ビット・データのトランスファを行うロジック



を出し続けます．このような状態を抜け出す方法は，通常はシステムの電源をいったんOFFし，再度ONする以外にありません．

しかし，図72のような回路を付加することにより，前記のような状態から抜け出すことができます．この回路はバス・リクエスト・フリップフロップがセットされていないときには，Z80 DMAの$\overline{BUSRQ}$信号がCPUに伝わらないようにしています．このフリップフロップはDMA動作を行いたいときに，プログラムによりセットし，リセット信号によりクリヤされます．

したがって，前記のような状態になったときリセット信号さえ出力することにより，このフリップフロップは強制的にクリヤされますから，CPUに対する$\overline{BUSRQ}$は"H"となります．

$\overline{BUSRQ}$が"H"であると，CPUは$\overline{BUSAK}$を"H"にします．この$\overline{BUSAK}$が"H"になると，Z80 DMAはバス・マスタであることを停止し，バスをCPUに返します．この状態のときには，Z80 DMAはコマンドを受け付けますので，リセット・コマンド（$C3_{16}$）を出すことにより，Z80 DMAを完全にリセットすることができます．

16ビット・データの転送

一般的にいって，DMAコントローラというのは，アドレスとメモリ，およびI/Oに対するリード／ライト信号を発生するのが役目ですので，転送するデータの幅（ビット数）には関知しません．しかし，Z80 DMAの場合にはリード・データを内部に一時ラッチした後にライト動作に入ります（リード・サイクルとライト・サイクルが完全に分かれている）．

したがって，Z80 DMAを用いて8ビット以上の幅のデータ転送を行うには，外部にデータ・ラッチを必要とします．図73に16ビット・データを転送するための構成を示します．Z80 DMAはリード・サイクルの最後のクロックの立ち下がり時に，データ・バス上のデータを内部にラッチし，ライト・サイクル時にこのデータをバス上に乗せます．

〈図74〉 同時データ転送ロジックを説明するための，信号名の付けかた

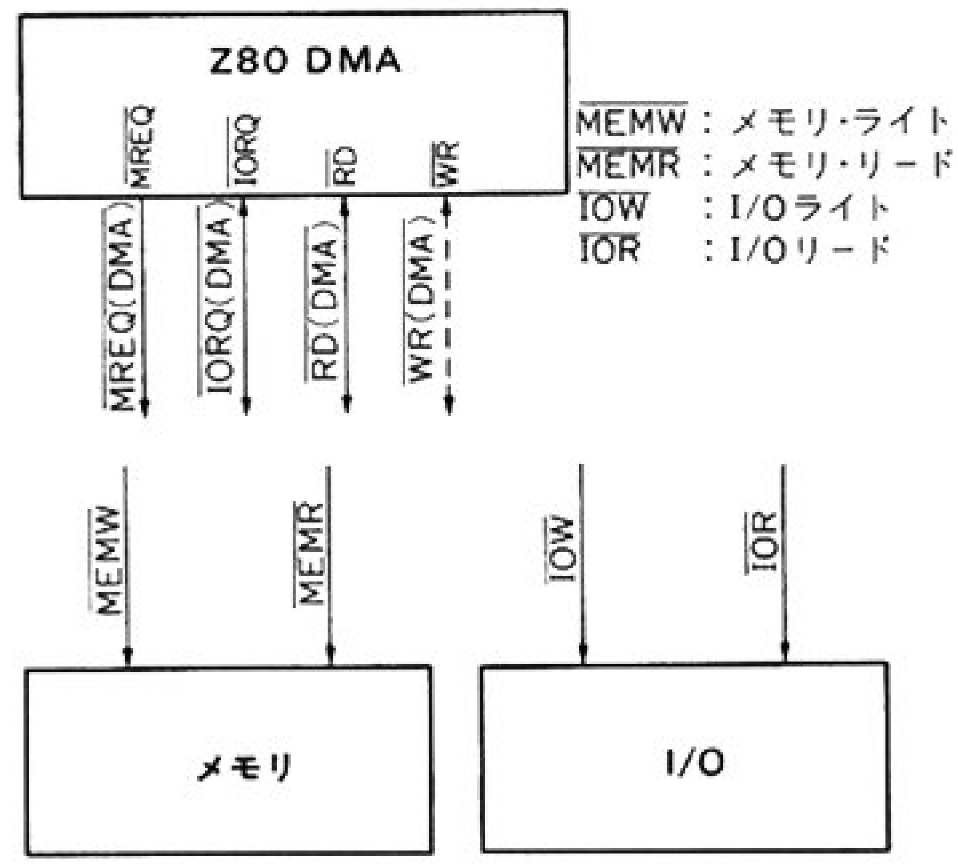


図73のラッチ（LS374）はデータの上位8ビット（D_8～D_{15}）を，$\overline{RD}$信号が"L"であるときのCLKの立ち下がり時にラッチし，$\overline{RD}$信号がなくなるとそのデータを，バス上に出力します．なお，図73のロジックにより，16ビット・データのトランスファは可能ですが，サーチはできません．

同時データ転送

同時データ転送というのは，Z80 DMAにはメモリ・サーチ，またはI/Oサーチ動作をやらせておき，外部ロジックにより，メモリとI/Oに対する制御信号を発生し，1リード・サイクル中（サーチ）にリードおよびライト動作を行ってしまうテクニックです．もちろん，その目的はデータ転送速度の向上にあります．

図74に同時データ転送のロジックを説明するための信号名の付けかたを示します．同時データ転送では，サーチ・タイミングを利用しますので，Z80 DMAは$\overline{WR}$を出力しません．したがって，$\overline{WR}$を考慮の外におくため，図74では$\overline{WR}$を破線で示してあります．

図75に同時データ転送における，各信号間の真理値表を示します．この図からわかるように，メモリ・サーチ・タイミングを使おうと，I/Oサーチ・タイミングを使おうと，外部ロジックが発生すべき信号に差違はありませんが，データの転送方向にしたがって，メモリとI/Oに対する制御信号を変化させる必要があります．

どちらのタイミングを用いた場合でも，Z80 DMAは$\overline{RD}$信号を出力しますので，これを用いて他の信号を発生することができます．

図76に同時データ転送を行うための，メモリとI/O

に対するリード／ライト信号を発生するロジックを示します．このロジックはマルチプレクサにより構成され，同時データ転送マルチプレクサ（Simultaneous Transfer Multiplexer）と呼ばれます．このマルチプレクサの出力は $\overline{\text{BUSAK}}$ によりイネーブルされ，転送方向信号にしたがってメモリとI/Oに対する制御信号を発生します．

図77に**図76**のロジックを組み込んだ，同時データ転送用の構成を示します．このロジックでは，データ転送方向は「転送方向フリップフロップ」の出力により制御されます．この転送方向フリップフロップは，プログラムによりセットまたはリセットされ，セットされると転送方向はI/O→メモリとなり，リセットされるとメモリ→I/Oとなります．

同時データ転送では，Z80 DMAが出力するアドレスはメモリ・アドレスであると解釈しますので，I/Oアドレスという概念はなくなります．したがって，なんらかの方法により，I/Oのチップ・セレクト信号を作る必要がありますが，この例ではアドレス・デコーダの出力と $\overline{\text{BUSAK}}$ をORすることにより，DMAデータ転送時にもチップ・セレクト信号が出るようにしてあります．

メモリおよびI/Oには，同時データ転送マルチプレクサの出力のほかに，通常の制御信号（$\overline{\text{MREQ}}$，$\overline{\text{IORQ}}$，$\overline{\text{RD}}$，$\overline{\text{WR}}$）も入力されています．メモリおよびI/Oは，

〈図75〉 同時データ転送における各信号の真理値表

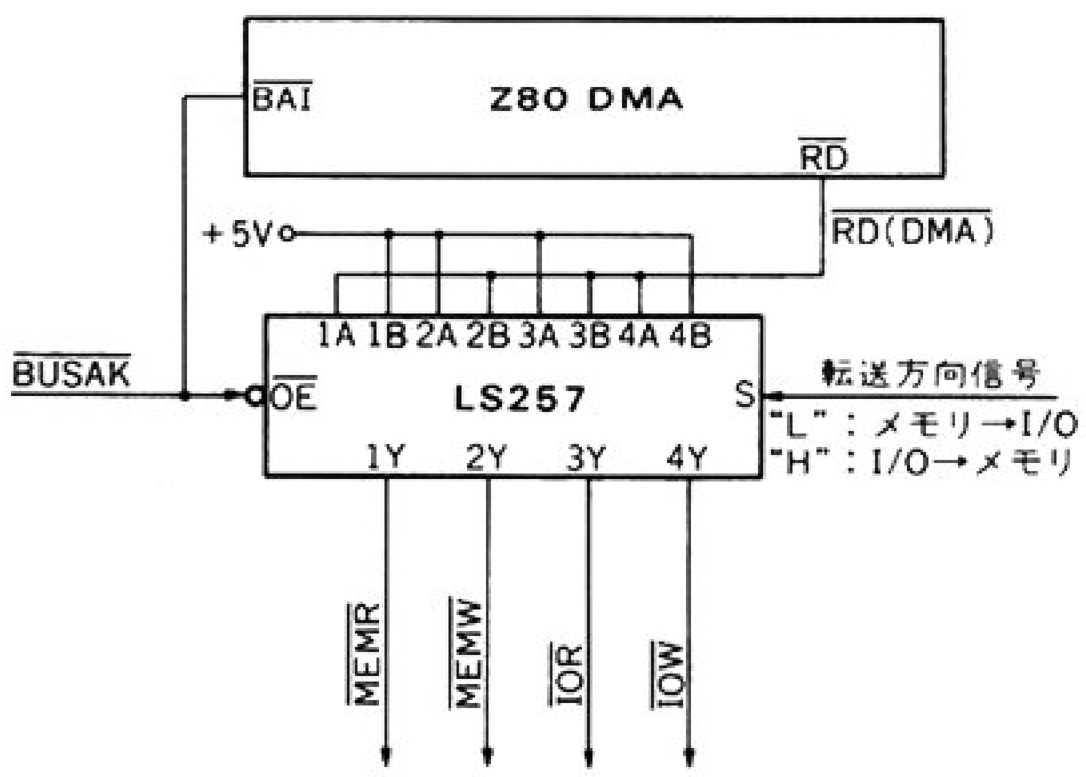
信号＼転送方向	Z80DMAが出力する信号		外部ロジックが発生する信号			
	$\overline{\text{MREQ}}$(DMA)	$\overline{\text{RD}}$(DMA)	$\overline{\text{MEMR}}$	$\overline{\text{MEMW}}$	$\overline{\text{IOR}}$	$\overline{\text{IOW}}$
メモリ→I/O	L	L	L	H	H	L
I/O→メモリ	L	L	H	L	L	H

(a) メモリ・サーチ・タイミングを用いたとき

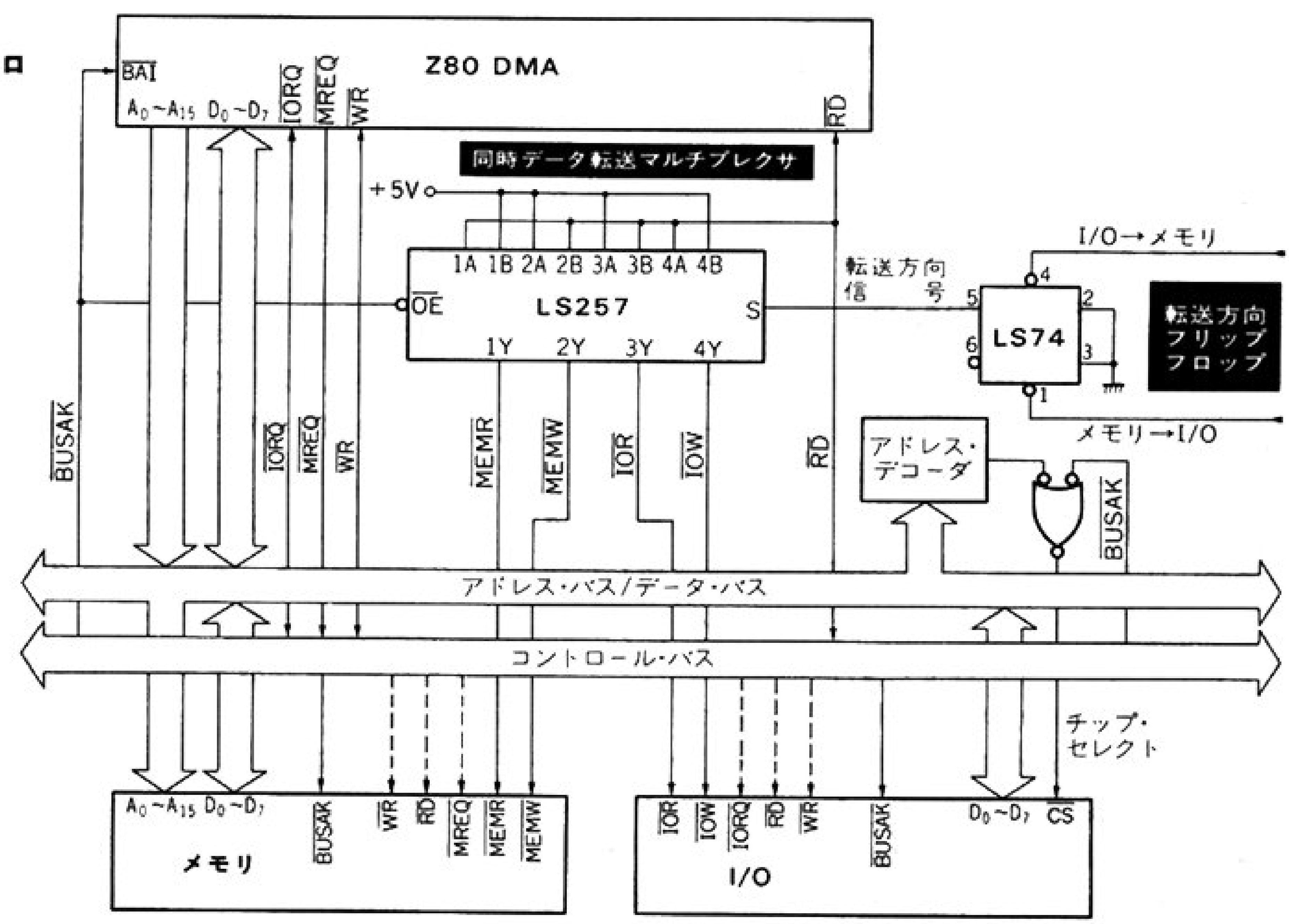
信号＼転送方向	Z80DMAが出力する信号		外部ロジックが発生する信号			
	$\overline{\text{IORQ}}$(DMA)	$\overline{\text{RD}}$(DMA)	$\overline{\text{MEMR}}$	$\overline{\text{MEMW}}$	$\overline{\text{IOR}}$	$\overline{\text{IOW}}$
メモリ→I/O	L	L	L	H	H	L
I/O→メモリ	L	L	H	L	L	H

(b) I/Oサーチ・タイミングを用いたとき

〈図76〉 同時データ転送マルチプレクサ



〈図77〉 同時データ転送を行うロジック



$\overline{\text{BUSAK}}$の極性にしたがって制御信号を切り替えることにより，CPUがバス・マスタにあるときにも，DMAがバス・マスタにあるときにも，データ転送を行うことができます．

Z80 DMAのプログラミング

ここでは，Z80 DMAを実際に動作させるプログラム例をいくつか示し，個々のプログラムの内容および派生的なことがらを説明します．

■Z80 DMAのリセット

前述のようにZ80 DMAがどのような状態にあっても，これを完全にリセットするにはリセット・コマンド（$C3_{16}$）を続けて6回出す必要があります．Prog❺にZ80 DMAをリセットするルーチンを示します．これ以降に示すプログラムには，すべてこのプログラムと同じルーチンが組み込まれています．

■Z80 DMAのリード・レジスタの読み取り

Z80 DMAにはリード・レジスタが七つ（RR0～RR6）あります．ここではこれらのレジスタの読み取り方法を示します．Z80 DMAにはRR0（ステータス・レジスタ）だけを読む方法と，RR0～RR6を読む方法とが用意されています．

・RR0だけを読む場合

❶ リード・ステータス・バイト・コマンド（BF_{16}）を出す

❷ リード命令を出す

・RR0～RR6を読む場合

❶ リード・マスク・フォローズ・コマンド（BB_{16}）を出す

❷ リード・マスクをライトする

❸ イニシエイト・リード・シーケンス・コマンド（$A7_{16}$）を出す

❹ リード命令をマスクの数だけ出す

Prog❻にリード・レジスタをリードするプログラムを示します．このプログラムには，RR0だけを読む場合，RR0～RR6を全部読む場合，それにRR0～RR6のうちのいくつかを読む場合が示されています．

メモリ ➡ メモリ・データ転送

Z80 CPUには，LDIRというメモリ間のブロック・データ転送用の命令がありますが，Z80 DMAのメモリ ➡ メモリ・データ転送機能を用いると，さらに高速にデータ転送ができます．Prog❼にZ80 DMAを用いて，メモリ ➡ メモリ・データ転送を行うプログラムを示します．

このプログラムは，0000_{16}番地から始まる1024バイトのメモリ・データを2400_{16}番地から始まるメモリ領域に移します．データ転送が終わると，Z80 DMAのリード・レジスタを読み取り，ついで転送されたデータと元のメモリ・データを比較し，データ転送が正しく行われたか否かをチェックします．

その後プログラムは，デスティネーション・バッファをクリヤし，再度DMAを起動します．アドレスおよびバイト・カウントを変更することなく，再度DMAを起動するときには，コマンド・チェインを始めから全部送る必要はなく，ロード・コマンド（CF_{16}），フォース・レディ（$B3_{16}$），ならびにイネーブルDMAコマンド（87_{16}）だけですみます．

このプログラムでは，ポートAおよびポートBに対して可変タイミングを採用しており，両ポートとも2サイクルで動くようにしてあります．したがって，1バイトのデータ転送は4クロック期間で行われます．Z80 DMAのイニシャライズに要するクロック数は397となっています（$200B_{16}$番地から2012_{16}番地までの命

〈Prog❺〉 Z80 DMAをリセットするルーチン

```
                          ;
                          ;      DMA0      RESET Z80 DMA
                          ;
0018                      DMA    EQU       18H                  ;DMA ADDRESS
                          ;
0000      3E C3                  LD        A,0C3H               ;RESET COMMAND
0002      06 06                  LD        B,6                  ;LOOP COUNT
0004      D3 18           LOOP:  OUT       (DMA),A              ;RESET COMMAND TO DMA
0006      10 FC                  DJNZ      LOOP                 ;OUTPUT SIX TIMES ?
                                                                ;NO,LOOP
                          ;
                          ;      END OF RESET DMA
                          ;
                                 END
```

命の実行に要するクロック数)．よってnバイトの転送に要するクロック数は，397＋4nとなります．

LDIR命令によりnバイトのデータ転送を行うのに要するクロック数は，アドレスおよびカウントのセットアップを含めると，30＋21nとなります（最後のデータの転送は16クロックで行われるが，これを無視します）．

したがって次の方程式を解くことにより，LDIR命令を用いるのと，Z80 DMAを用いるのと「どっちがトクかな？」の臨界点が求まります．

$$397+4n=30+21n$$

$$n=21.6$$

〈Prog❻〉 リード・レジスタを読むプログラム

```
                        ;
                        ;       Z80 DMA READ REGISTER READ
                        ;       THIS PROGRAM SHOWS HOW TO RAED THE READ
                        ;       REGISTERS OF Z80 DMA.
                        ;
0018                    DMA     EQU     18H             ;DMA ADDRESS
                        ;
                        ;       READ RR0(STATUS BYTE) ONLY
0000    3E BF                   LD      A,0BFH          ;READ STATUS BYTE COMAND
0002    D3 18                   OUT     (DMA),A         ;LOAD IT TO DMA
0004    DB 18                   IN      A,(DMA)         ;GET STATUS BYTE
                        ;
                        ;
                        ;       READ RR0-RR6
0006    3E BB                   LD      A,0BBH          ;READ MASK FOLOWS CMD
0008    D3 18                   OUT     (DMA),A         ;LOAD IT TO DMA
000A    3E 7F                   LD      A,07FH          ;SET READ MASK
000C    D3 18                   OUT     (DMA),A         ;LOAD IT TO DMA
000E    3E A7                   LD      A,0A7H          ;INITIATE READ SEQUENCE
0010    D3 18                   OUT     (DMA),A         ;LOAD IT TO DMA
0012    06 07                   LD      B,7             ;SET COUNT=7
0014    0E 18                   LD      C,DMA           ;SET DMA ADDRESS
0016    21 001D                 LD      HL,STATUS       ;SET BUFFER ADDRESS
0019    ED B2                   INIR                    ;READ RR0-RR6
001B    18 07                   JR      NEXT            ;
                        ;
                        ;       BUFFER
001D    00              STATUS: DB      0               ;STATUS BYTE
001E    00                      DB      0               ;BYTE COUNTER (LOWER)
001F    00                      DB      0               ;BYTE COUNTER (UPPER)
0020    00                      DB      0               ;PORT A ADDRESS (LOWER)
0021    00                      DB      0               ;PORT A ADDRESS (UPPER)
0022    00                      DB      0               ;PORT B ADDRES (LOWER)
0023    00                      DB      0               ;PORT B ADDRESS (UPPER)
                        ;
                        ;       READ RR0,RR3,RR4
0024    3E BB           NEXT:   LD      A,0BBH          ;READ MASK FOLLOWS
0026    D3 18                   OUT     (DMA),A         ;LOAD IT TO DMA
0028    3E 19                   LD      A,19H           ;SET READ MASK
002A    D3 18                   OUT     (DMA),A         ;LOAD IT TO DMA
002C    3E A7                   LD      A,0A7H          ;INITIATE READ SEQUENCE
002E    D3 18                   OUT     (DMA),A         ;LOAD IT TO DMA
0030    06 03                   LD      B,3             ;SET COUNT
0032    0E 18                   LD      C,DMA           ;SET DMA ADDRES
0034    21 003B                 LD      HL,STAT         ;SET BUFFER ADDRESS
0037    ED B2                   INIR                    ;READ RR0,RR3,RR4
0039    18 FE                   JR      $               ;
                        ;
                        ;       BUFFER
003B    00              STAT:   DB      0               ;STATUS BYTE
003C    00                      DB      0               ;PORT A ADDRESS (LOWER)
003D    00                      DB      0               ;PORT A ADDRESS (UPPER)
                        ;
                        ;
                                END
```

これから，22バイト以上のデータ転送を行うときには，Z80 DMA を用いたほうが速くなることがわかります（図13参照——207ページ）．

Prog ❼ では Z80 DMA をバースト・モードに設定してありますが，コンティニュアス・モードに設定しても動作は同じです．ただし，バイト・モードと設定すると，1バイトのデータを転送し終わったときに，フォース・レディ状態が解除されてしまいますので，このままでは動きません．

LDIR 命令がメモリ・クリヤに使われるのと同じように，Z80 DMA のメモリ → メモリ・データ転送を利用して，メモリ・クリヤを行うこともできます．

Prog ❼ では，1回のブロック転送が終わるごとにDMAのリード・レジスタを読み取ります．**図78**にDMAデータ転送が終結したときの，リード・レジスタの値を示します．**図39**（228ページ）と見比べてみてください．

メモリ・サーチ

Prog ❽ に Z80 DMA によるメモリ・サーチのプログラムを示します．このプログラムは，キャラクタ・ストリング（SRCE）中の文字，“？”を捜し出します．メモリ・サーチが終了したときの，リード・レジスタの値は**図79**のようになります．

理由はわかりませんが，ステータス・バイトのビット0（DMA動作が行われた）が“1”になりません．またバイト・カウンタの値も**図40**とは異なります．

Prog ❽ のマッチ・バイトを FF_{16} に変えてみた場合の，DMA動作終了時のリード・レジスタの値を**図80**に示します．この場合にはマッチは検出されず，エン

〈Prog ❼〉 メモリ → メモリ・データ転送

```
                        ;
                        ;       DMA1    Z80 DMA MEMORY TO MEMORY TEST
                        ;       THIS PROGRAM TRANSFERS 1024 BYTES OF DATA
                        ;       ADDRESSED AT 0000H TO DEST. BUFFER (2400H) AND
                        ;       VERIFYS TRANSFERED DATA FOR VALIDTY.
                        ;
0018                    DMA     EQU     18H             ;DMA ADDRESS
                        ;
0000                    SRCE    EQU     0000H           ;SOURCE ADDRESS
2400                    DEST    EQU     2400H           ;DESTINATION ADDRESS
03FF                    LENGTH  EQU     1024-1          ;TRANSFER LENGTH (1024)
                        ;
0000                            ASEG
                                ORG     2000H
2000    31 20AC         BEGIN:  LD      SP,STACK        ;SET STACK POINTER
                        ;       RESET DMA
2003    3E C3                   LD      A,0C3H          ;LOAD RESET COMMAND
2005    06 06                   LD      B,6             ;LOAD LOOP COUNT
2007    D3 18           LOOP:   OUT     (DMA),A         ;RESET COMMAND TO DMA
2009    10 FC                   DJNZ    LOOP            ;OUTPUT SIX TIMES ?
                                                        ;NO,LOOP
                        ;
200B    21 2052                 LD      HL,DMACC        ;LOAD DMA COMMAND ADRS
200E    06 12                   LD      B,DMACL-DMACC   ;LOAD COMMAND LENGTH
2010    0E 18           BEGINO: LD      C,DMA           ;LOAD DMA ADDRESS
2012    ED B3                   OTIR                    ;LOAD COMMAND TO DMA
                        ;
                        ;       NOW READ DMA STAUTS
2014    3E BB                   LD      A,0BBH          ;READ BYTE FOLLOWS
2016    D3 18                   OUT     (DMA),A         ;
2018    3E 7F                   LD      A,7FH           ;SET ALL INFORMATIONS
201A    D3 18                   OUT     (DMA),A         ;
201C    3E A7                   LD      A,0A7H          ;INITIATE READ SEQUENCE
201E    D3 18                   OUT     (DMA),A         ;LOAD IT TO DMA
2020    06 07                   LD      B,7             ;SET LENGTH
2022    0E 18                   LD      C,DMA           ;LOAD DMA ADDRESS
2024    21 2064                 LD      HL,STATUS       ;LOAD STAYUS BUF ADRS
2027    ED B2                   INIR                    ;READ STATUS FROM DMA
                        ;       NOW CHECK DATA TRANSFERED
2029    11 0000                 LD      DE,SRCE         ;LOAD SOURCE ADDRESS
202C    21 2400                 LD      HL,DEST         ;LOAD DESTINATION ADRS
202F    01 0400                 LD      BC,LENGTH+1     ;LOAD BLOCK LENGTH
```

(次ページへつづく)

```
2032    1A             CHECK:   LD      A,(DE)          ;LOAD SOURCE DATA
2033    BE                      CP      (HL)            ;DATA MATCH ?
2034    20 1B                   JR      NZ,ERROR        ;NO,ERROR
2036    13                      INC     DE              ;BUMP SOURCE POINTER
2037    23                      INC     HL              ;BUMP DESTINATION PTR
2038    0B                      DEC     BC              ;COUNT DOWN LENGTH
2039    78                      LD      A,B             ;CHECKED ALL ?
203A    B1                      OR      C               ;
203B    20 F5                   JR      NZ,CHECK        ;NO,CHECK
                       ;        CLEAR DESTINATION BUFFER
203D    21 2400                 LD      HL,DEST         ;LOAD DESTINATION ADRS
2040    11 2401                 LD      DE,DEST+1       ;
2043    01 03FF                 LD      BC,LENGTH       ;LENGTH TO CLEAR
2046    36 FF                   LD      (HL),0FFH       ;FILL WITH FF'S
2048    ED B0                   LDIR                    ;DO FILLING
                       ;        NOW RELOAD ADDRESS AND FORCE READY
204A    21 2061                 LD      HL,DMACCO       ;LOAD COMMAND ADDRESS
204D    06 03                   LD      B,DMACL-DMACCO  ;LOAD COMMAND LENGTH
204F    18 BF                   JR      BEGINO          ;TRY AGAIN !!!
                       ;
2051    76             ERROR:   HALT                    ;DATA ERROR STOP
                       ;
                       ;        DMA COMMAND CHAIN
2052    C3             DMACC:   DB      0C3H     ;WR6   RESET COMMAND
2053    7D                      DB      7DH      ;WR0   TRANSFER
                                                 ;      PORT A --> PORT B
                                                 ;      PORT A ADRS FOLLOWS
                                                 ;      PORT A LENGTH FOLLOWS
2054    0000                    DW      SRCE     ;      PORT A ADDRESS
2056    03FF                    DW      LENGTH   ;      PORT A LENGTH
2058    54                      DB      54H      ;WR1   PORT A ADRS INCREMENTS
                                                 ;      PORT A IS MEMORY
                                                 ;      TIMING BYTE FOLLOWS
2059    02                      DB      02H      ;      TIMING BYTE
                                                 ;      TWO CYCLE EARLY END
205A    50                      DB      50H      ;WR2   PORT B ADRS INCREMENTS
                                                 ;      PORT B IS MEMORY
                                                 ;      TIMING BYTE FOLLOWS
205B    02                      DB      02H      ;      TIMING BYTE
                                                 ;      TWO CYCLE EARLY END
205C    80                      DB      80H      ;WR3   INTERRUPT DISABLE
205D    CD                      DB      0CDH     ;WR4   BURST MODE
                                                 ;      PORT B ADRS FOLLOWS
205E    2400                    DW      DEST     ;      PORT B ADDRESS
2060    8A                      DB      8AH      ;WR5   STOP ON END OF BLOCK
                                                 ;      CE/ ONLY
                                                 ;      READY ACTIVE HIGH
2061    CF             DMACCO:  DB      0CFH     ;WR6   LOAD COUNT & ADDRESS
2062    B3                      DB      0B3H     ;WR6   FORCE READY
2063    87                      DB      87H      ;WR6   ENABLE DMA
2064                   DMACL    EQU     $               ;
                       ;
                       ;        DMA STATUS WILL BE AS FOLLOWS
                       ;        RR0     0DFH     END OF BLOCK, DMA OCCURRED
                       ;        RR1     0FFH     BYTE COUNTER (LOW)
                       ;        RR2     03H      BYTE COUNTER (HIGH)
                       ;        RR3     00H      PORT A ADDRESS (LOW)
                       ;        RR4     04H      PORT A ADDRESS (HIGH)
                       ;        RR5     0FFH     PORT B ADDRESS (LOW)
                       ;        RR6     27H      PORT B ADDRESS (HIGH)
                       ;
2064                   STATUS:  DS      8               ;STATUS BUFFER
                       ;
206C                            DS      64              ;STACK AREA
20AC                   STACK    EQU     $               ;
                       ;
                                END
```

レジスタ	レジスタの内容	値 (16進数)	説　　明
RR0	ステータス・レジスタ	DF	・エンド・オブ・ブロック ・レディ・ラインは無効極性 ・DMA動作が行われた
RR1, RR2	バイト・カウンタ	03FF	N
RR3, RR4	ポートAアドレス・カウンタ	0400	As +(N+1)
RR5, RR6	ポートBアドレス・カウンタ	27FF	As' +N

〈図78〉
Prog ⑦ の DMA データ転送終了時のリード・レジスタの値

$\left(\begin{array}{l} As=0000_{16} \\ As'=2400_{16} \\ N=03FF_{16} \end{array}\right)$

レジスタ	レジスタの内容	値 (16進数)	説　　明
RR0	ステータス・バイト	AE	・マッチ検出 ・レディ・ラインは無効レベル ・DMA動作は行われなかった(?)
RR1, RR2	バイト・カウンタ	000C	M+1(?)
RR3, RR4	ポートAアドレス・カウンタ	204A	As+(M+1)

〈図79〉
Prog ⑧ の動作終了時（マッチ検出）のリード・レジスタの値

$\left(\begin{array}{l} M(\text{マッチ・バイトのバイト番号}) \\ \qquad =11_{10} \\ As=203E_{16} \end{array}\right)$

ド・オブ・ブロックにより，DMA動作は終結します．このときにも，ステータス・バイトのビット0は"1"にならず，バイト・カウンタの値も図39とは1だけ差があります．

Z80 CPU には，CPIR というメモリ・サーチ用の命令があります．この命令では，nバイト目にマッチが検出されたとすると，必要なクロック数はレジスタのセットアップを含めて，27+21n となります．

Prog ⑧ のように Z80 DMA を用いてメモリ・サーチを行った場合には，Z80 DMA のイニシャライズ部分を含めると，必要なクロック数は 355+2n となります．

メモリ間データ転送と同じように両者の臨界点を求めますと，

$$27+21\ n=355+2\ n$$
$$n=17.3$$

となり，18バイトを越えると，Z80 DMA を用いたほうがだんぜん速くなります（図12参照——206ページ）．ただし，Prog ⑧ では可変サイクルを用い，メモリ・サイクルを2サイクルと設定しています．

なお申し遅れましたが，一番最後のデータでマッチが検出されたときには，ステータス・バイトにはエンド・オブ・ブロックとマッチの両方がセット（"0"）されます．

メモリ → I/O データ転送

図81にメモリから I/O (Z80 PIO) へ，DMA データ転送するための構成を示します．PIO はポートAをモード0（出力モード）に設定し，ARDY を Z80 DMA の RDY に接続します．Z80 DMA の RDY ラインの有効極性は "L" とします．PIO はモード0にプログラミングされると，ARDY は "L" になりますので，DMA に対する RDY は有効極性になります．

Z80 DMA はイネーブルされると，メモリから I/O (PIO ポートA) へデータを転送します．データを受け取ると，PIO ポートAは ARDY を "H" にします．

ARDY が "H" になると，外部装置は PA_0～PA_7 上のデータを取り込み，$\overline{ASTB}$ を入力します．これにより再度 ARDY は "L" になります．ARDY が "L" になると，Z80 DMA は次のデータをメモリから読み取り，これを PIO に送ります．

以上の過程を行うプログラムを，Prog ⑨ に示します．このプログラムはエンド・オブ・ブロック時に割り込みがかけられ，割り込みルーチン (I02) はステータス・バイトをイニシャライズした後，再度ロード・コマンドによりアドレス・カウンタにアドレスをロードし，DMA をイネーブルします．

I/O → メモリ・データ転送

図82に I/O (PIO ポートB) からメモリへ，DMA データ転送するための構成を示します．PIO のポートBはモード1（入力）に設定されます．PIO の BRDY を Z80 DMA のRDY に接続します．PIO のポートBはモード1に設定されると，BRDY を"H"にします(バッファ・エンプティ)．したがって，Z80 DMA の RDY ラインの有効極性は "L" とします．

〈Prog ⑧〉 メモリ・サーチ

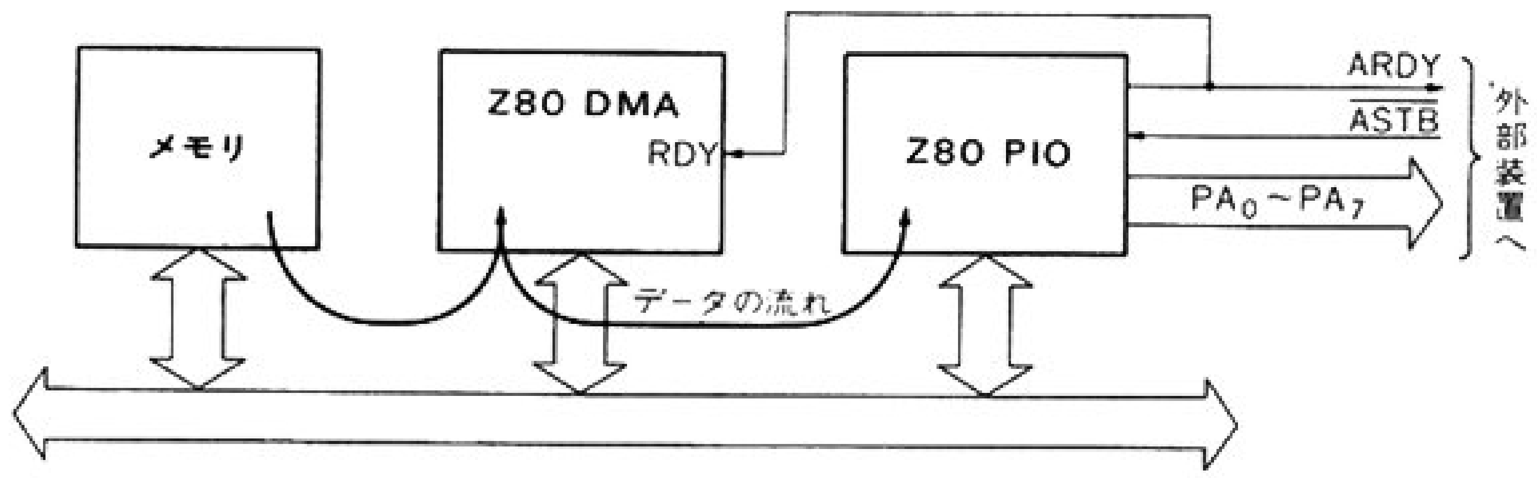
```
                        ;
                        ;       DMA2     Z80 DMA MEMORY SEARCH TEST
                        ;
0018                    DMA     EQU      18H             ;DMA ADDRESS
                        ;
0000                            ASEG
                                ORG      2000H
2000    31 20A3         BEGIN:  LD       SP,STACK        ;SET STACK POINTER
                        ;       RESET DMA
2003    3E C3                   LD       A,0C3H          ;LOAD RESET COMMAND
2005    06 06                   LD       B,6             ;SET COUNT
2007    D3 18           LOOP:   OUT      (DMA),A         ;RESET COMMAND TO DMA
2009    10 FC                   DJNZ     LOOP            ;LOADED SIX TIMES ?
                                                         ;NO,LOOP
200B    21 2026                 LD       HL,DMACC        ;LOAD DMA COMMAND ADRS
200E    06 10                   LD       B,DMACL-DMACC   ;LOAD COMMAND LENGTH
2010    0E 18                   LD       C,DMA           ;LOAD DMA ADDRESS
2012    ED B3                   OTIR                     ;LOAD COMMAND TO DMA
                        ;       NOW READ DMA STATUS
2014    3E BB                   LD       A,0BBH          ;READ BYTE FOLLOWS
2016    D3 18                   OUT      (DMA),A         ;
2018    3E 7F                   LD       A,7FH           ;SET ALL INFORMATIONS
201A    D3 18                   OUT      (DMA),A         ;
201C    06 07                   LD       B,7             ;SET LENGTH=7
201E    0E 18                   LD       C,DMA           ;SET DMA ADDRESS
2020    21 2036                 LD       HL,STATUS       ;LOAD STATUS BUF ADDRESS
2023    ED B2                   INIR                     ;READ DMA STATUSES
2025    76                      HALT                     ;
                        ;
                        ;       DMA COMMAND CHAIN
2026    C3              DMACC:  DB       0C3H      ;WR6   RESET DMA
2027    7E                      DB       7EH       ;WR0   SEARCH ONLY
                                                   ;      PORT A --> PORT B
                                                   ;      PORT A ADRS FOLLOWS
                                                   ;      PORT A LENGTH FOLLOWS
2028    203E                    DW       SRCE      ;      PORT A ADRS
202A    0024                    DW SRCEL-SRCE-1    ;      LENGTH
202C    54                      DB       54H       ;WR1   PORT A ADRS INCREMENTS
                                                   ;      PORT A IS MEMORY
                        ;                          ;      TIMING BYTE FOLLOWS
202D    02                      DB       02H       ;      TIMING BYTE
                                                   ;      TWO CYCLE EARLY END
202E    9C                      DB       9CH       ;WR3   INTERRUPT DISABLE
                                                   ;      MASK & MATCH BYTE FOLLOW
                                                   ;      STOP ON MATCH
202F    00                      DB       00H       ;      MASK BYTE (NO MASK)
2030    3F                      DB       '?'       ;      MATCH BYTE
2031    C1                      DB       0C1H      ;WR4   BURST MODE
2032    8A                      DB       8AH       ;WR5   STOP ON END OF BLOCK
                                                   ;      CE/ ONLY
                                                   ;      READY ACRIVE HIGH
2033    CF                      DB       0CFH      ;WR6   LOAD ADRS & COUNT
2034    B3                      DB       0B3H      ;WR6   FORCE READY
2035    87                      DB       87H       ;WR6   ENABLE DMA
2036                    DMACL   EQU      $               ;
                        ;
2036                    STATUS: DS       8               ;STATUS BUFFER
                        ;
                        ;       DATA STRING FOR SEARCH
203E    30 31 32 33     SRCE:   DB       '0123'          ;
2042    34 35 36 37             DB       '4567'          ;
2046    38 39                   DB       '89'            ;
2048    3F                      DB       '?'             ;MATCH BYTE
2049    41 42 43 44             DB       'ABCD'          ;
204D    45 46 47 48             DB       'EFGH'          ;
```

```
2051      49 4A 4B 4C              DB      'IJKL'         ;
2055      4D 4E 4F 50              DB      'MNOP'         ;
2059      51 52 53 54              DB      'QRST'         ;
205D      55 56 57 58              DB      'UVWX'         ;
2061      59 5A                    DB      'YZ'           ;
2063                     SRCEL     EQU     $              ;
                         ;
2063                               DS      64             ;STACK AREA
20A3                     STACK     EQU     $              ;
                         ;
                                   END
```

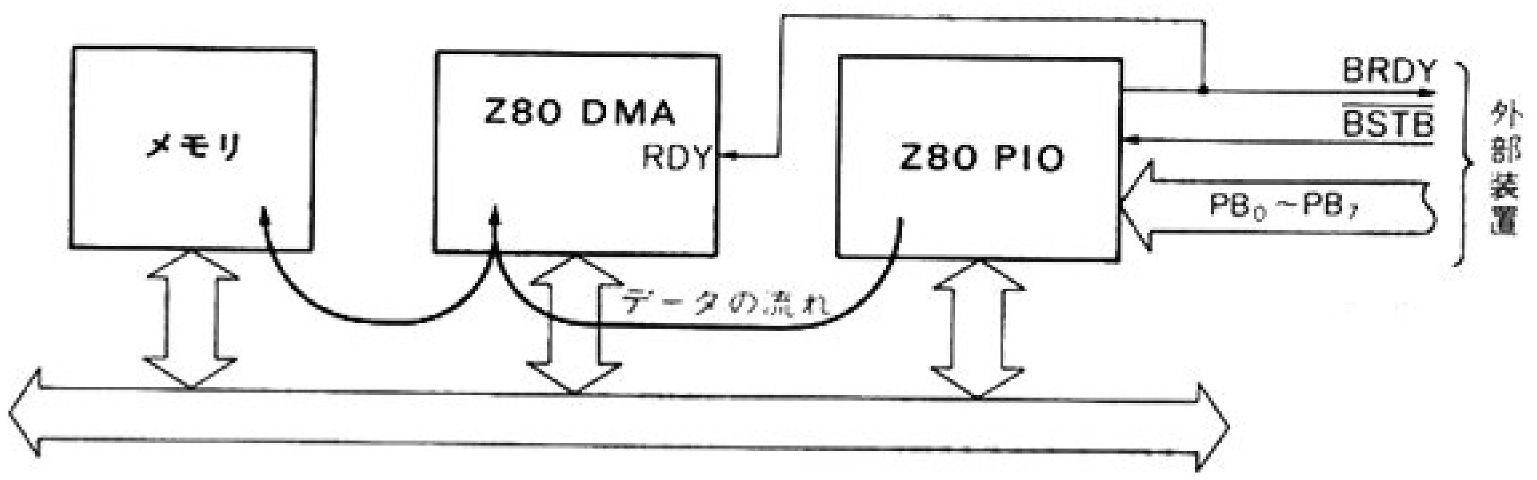
レジスタ	レジスタの内容	値 (16進数)	説明
RR0	ステータス・レジスタ	9E	・エンド・オブ・ブロック ・レディ・ラインは無効極性 ・DMA動作は行われなかった
RR1, RR2	バイト・カウンタ	0025	N+1(?)
RR3, RR4	ポートAアドレス・カウンタ	2063	As+(N+1)

〈図80〉
Prog ⑧ の DMA 動作がエンド・オブ・ブロックで終結したときの，リード・レジスタの値

$\begin{pmatrix} N=24_{16} \\ As=203E_{16} \end{pmatrix}$



〈図81〉
Prog ⑨ が動作する構成
メモリ → I/O (PIO ポートA)



〈図82〉
Prog ⑩ が動作する構成．I/O (PIO ポートB) → メモリ

外部装置が PB_0～PB_7 上にデータを乗せ，$\overline{BSTB}$ を印加すると，その立ち上がりで ARDY は "L" になります．これにより，Z80 DMA は PIO のポートBからデータを読み取り，これをメモリに書き込みます．データが読み取られると，PIO の BRDY は再び "H" になります．

Prog ⑩ に以上の過程を行うプログラムを示します．このプログラムはエンド・オブ・ブロックにより，割り込みがかけられ，割り込みルーチンではステータス・バイトをイニシャライズした後，ロード・コマンドおよびイネーブル DMA コマンドにより，再度 DMA をイネーブルします．

二つの Z80 DMA による PIO のループ・バック

図83に2個の Z80 DMA と PIO のループ・バック・テストを行う構成を示します．

各 Z80 DMA には，DMA1 および DMA2 という名前を付けます．このテストでは，まず DMA1 によりメモリから PIO のポートAにデータを転送します．ポートA (PA_0～PA_7) 上に現われたデータは，ループ・バック (Loop Back) され，PIO のポートB (PB_0～PB_7) にラッチされます．DMA2 はこのポートBにラッチされたデータをメモリに移します．

DMA1 の転送モードはバイト・モードで，データの

転送方向はメモリ → I/O とします．DMA2 の転送モードもやはりバイト・モードで，転送方向は I/O → メモリとします．PIO のポート A はモード 0（出力モード）と設定し，ポート B はモード 1（入力モード）と設定します．PIO のポート A の ARDY は DMA1 の RDY に接続され，ポート B の BRDY は DMA2 の RDY に接続されます．

PIO はモード 0 に設定されると，RDY は "L" となり，データがポートにライトされると "H" になりますので，ARDY が接続された DMA1 の RDY ラインの有効極性は "L" とします．

また，PIO はモード 1 に設定されると，RDY は "H" になり，ポートがデータをラッチすると，"L" になります．したがって，BRDY が接続された DMA2 の RDY ラインの有効極性もやはり "L" とします．

この構成ではまず DMA1 がメモリからのデータを

〈Prog ❾〉 メモリ → I/O

```
                      ;
                      ;       DMA4    MEMORY TO IO DMA DATA TRANSFER
                      ;       THIS PROGRAM TRANSFERS MEMORY DATA TO PIO
                      ;       VIA Z80 DMA. DMA READS DATA FROM MEMORY AND
                      ;       WRITES IT INTO PIO PORT A.
                      ;
0014                  PIOAD   EQU     14H             ;PIO CHANNEL A DATA
0015                  PIOAC   EQU     15H             ;PIO CHANNEL A CONTROL
                      ;
0018                  DMA     EQU     18H             ;DMA ADDRESS
                      ;
0000                  SRCE    EQU     0               ;SOURCE MEMORY ADDRESS
0100                  LENGTH  EQU     256             ;TRANSFER LENGTH
                      ;
                      ;
0000                          ASEG
                              ORG     2000H
2000    F3            BEGIN:  DI                      ;DISABLE INTERRUPT
2001    31 2138               LD      SP,STACK        ;SET STACK POINTER
2004    3E 20                 LD      A,20H           ;SET IR
2006    ED 47                 LD      I,A             ;
2008    ED 5E                 IM      2               ;SET MODE 2 INTERRUPT
                      ;       PIO CHANNEL A INITIALIZATION
200A    3E 0F                 LD      A,0FH           ;SET MODE 0 (OUTPUT)
200C    D3 15                 OUT     (PIOAC),A       ;
200E    3E 07                 LD      A,07H           ;DISABLE INTERRUPT
2010    D3 15                 OUT     (PIOAC),A       ;
                      ;       RESET DMA
2012    3E C3                 LD      A,0C3H          ;LOAD RESET COMMAND
2014    06 06                 LD      B,6             ;SET COUNT
2016    D3 18         LOOP:   OUT     (DMA),A         ;RESET COMMAND TO DMA
2018    10 FC                 DJNZ    LOOP            ;LOADED SIX TIMES ?
                                                      ;NO,LOOP
                      ;       DMA INITIALIZATION
201A    06 12                 LD      B,DMACL-DMACC   ;LOAD COMMAND LENGTH
201C    0E 18                 LD      C,DMA           ;LOAD DMA ADDRESS
201E    21 2039               LD      HL,DMACC        ;LOAD COMMAND ADDRESS
2021    ED B3                 OTIR                    ;COMMAND TO DMA
2023    FB                    EI                      ;ENABLE INTERRUPT
2024    18 FE                 JR      $               ;LOOP ETERNALLY
                      ;
                      ;       INT ON READY INTERRUPT
2026    76            I00:    HALT                    ;SPURIOUS INTERRUPT
                      ;
                      ;       MATCH INTERRUPT
2027    76            I01:    HALT                    ;SPURIOUS INTERRUPT
                      ;
                      ;       END OF BLOCK INTERRUPT
2028    3E 8B         I02:    LD      A,8BH           ;RE-INITIALIZE STATUS
202A    D3 18                 OUT     (DMA),A         ;LOAD COMMAND TO DMA
202C    06 04                 LD      B,DMACL-DMACC0  ;LOAD COMMAND LENGTH
202E    0E 18                 LD      C,DMA           ;LOAD DMA ADDRESS
2030    21 2047               LD      HL,DMACC0       ;LOAD COMMAND ADDRESS
```

PIO のポートAにライトします．これにより ARDY が“H”になり，ワンショットをトリガします．このワンショットの出力はポートBの $\overline{\mathrm{BSTB}}$ に接続されており，その立ち上がり時点にポートA上のデータ (PA_0～PA_7) のポートBにラッチさせます．

ポートBはデータをラッチすると BRDY を“L”にします．この BRDY が“L”になると，DMA2 の RDY ラインが有効極性になり，DMA2 は PIO のポートBからデータを読み取り，それをメモリに書き込みます．

PIO のポートBのデータが読み取られると，BRDY は再度“H”になりますが，これによりポートAの $\overline{\mathrm{ASTB}}$ が立ち上がることになり，ポートAの ARDY は“L”になります．これにより，DMA1 の RDY ラインが有効極性となり，DMA1 は次のデータをメモリから PIO ポートAに転送します．図84にこの二つの DMA による，ループ・バック・テストのタイミングを示します．

DMA データ転送

```
2033    ED B3                    OTIR                          ;LOAD COMMAND TO DMA
2035    FB                       EI                            ;RE-ENABLE INTERRUPT
2036    ED 4D                    RETI                          ;
                        ;
                        ;        MATCH,END OF BLOCK INTERRUPT
2038    76              IO3:     HALT                          ;SPURIOUS INTERRUPT
                        ;
                        ;        DMA COMMAND CHAIN
                        ;        PORT A IS ASSIGNED TO PIO CHANNEL A
                        ;        PORT B IS ASSIGNED TO MEMORY
2039    C3              DMACC:   DB      OC3H       ;WR6       RESET COMMAND
203A    6D                       DB      6DH        ;WR0       TRANSFER
                                                    ;          PORT A --> PORT B(TEMP)
                                                    ;          ADDRESS FOLLOWS (LOWER)
                                                    ;          LENGTH FOLLOWS
203B    14                       DB      PIOAD      ;          PORT  A DESTINATION
203C    00FF                     DW   LENGTH-1      ;          BLOCK LENGTH
203E    2C                       DB      2CH        ;WR1       PORT A IS I/O
                                                    ;          ADDRESS FIXED
                                                    ;          NO TIMING BYTE FOLLOWS
203F    10                       DB      10H        ;WR2       PORT B IS MEMORY
                                                    ;          ADDRESS INCREMENTS
                                                    ;          NO TIMING BYTE FOLLOWS
2040    A0                       DB      0A0H       ;WR3       ENABLE INTERRUPT
2041    9D                       DB      9DH        ;WR4       PORT A ADRS FOLLOWS
                                                    ;          INT CTL BYTE FOLLOWS
                                                    ;          BYTE MODE TRANSFER
2042    0000                     DW      SRCE       ;          SOURCE ADDRESS
2044    32                       DB      32H        ;          INTERRUPT CONTROL BYTE
                                                    ;          INT AT END OF BLOCK
                                                    ;          STATUS AFFECTS VECTOR
                                                    ;          VECTOR FOLLOWS
2045    F0                       DB DAINTV-BEGIN    ;          INTERRUPT VECTOR
2046    82                       DB      82H        ;WR5       READY ACTIVE LOW
                                                    ;          CE/ ONLY
                                                    ;          STOP AT END OF BLOCK
2047    CF              DMACC0:  DB      0CFH       ;WR6       LOAD ADDRESS TO PORT A
2048    01                       DB      01H        ;WR0       PORT B --> PORT A
2049    CF                       DB      0CFH       ;WR6       LOAD ADDRESS TO PORT B
204A    87                       DB      87H        ;WR6       ENABLE DMA
204B                    DMACL    EQU     $
                        ;
                                 ORG     BEGIN+0F0H
                        ;        DMA INTERRUPT VECTOR
20F0    2026            DAINTV:  DW      IO0                   ;INT ON READY
20F2    2027                     DW      IO1                   ;MATCH
20F4    2028                     DW      IO2                   ;END OF BLOCK
20F6    2038                     DW      IO3                   ;MATCH,END OF BLOCK
                        ;
20F8                             DS      64                    ;STACK AREA
2138                    STACK    EQU     $
                        ;
                                 END
```

Prog ⑪に以上の過程を行うプログラムを示します．このプログラムにおける DMA1 および DMA2 に対するプログラムは，それぞれ Prog ⑨ および Prog ⑩ におけるそれと同じです．

このプログラムでは 256 バイトのデータを転送し終わると，DMA1 のエンド・オブ・ブロック割り込みが発生します．この割り込みルーチンでは，ステータス・バイトをイニシャライズして RETI します．DMA2 が最後のデータを転送し終わると，DMA2 のエンド・オブ・ブロック割り込みが発生します．

このルーチンでは PIO のポート A に送ったデータ (SRCE) とポート B からループ・バックしてきたデータが等しいかどうかをチェックした後に，データを 1 バイトだけシフトし，再度両 DMA をイネーブルします．

以上で Z80 DMA の解説を終ります．本項では先ず，DMA とは何か？　というタイトルで DMA の意味と必要性を説き，つぎに Z80 DMA の機能とプログラミング方法を説明し，最後にいくつかのプログラム例を示しました．以上の解説で Z80 DMA をすべて説明しきっているとは思いませんが，通常の応用では本項で述べた事柄を知っていれば十分であり，また Z80 DMA を使うにはこれくらいの知識は最低限必要なものです．

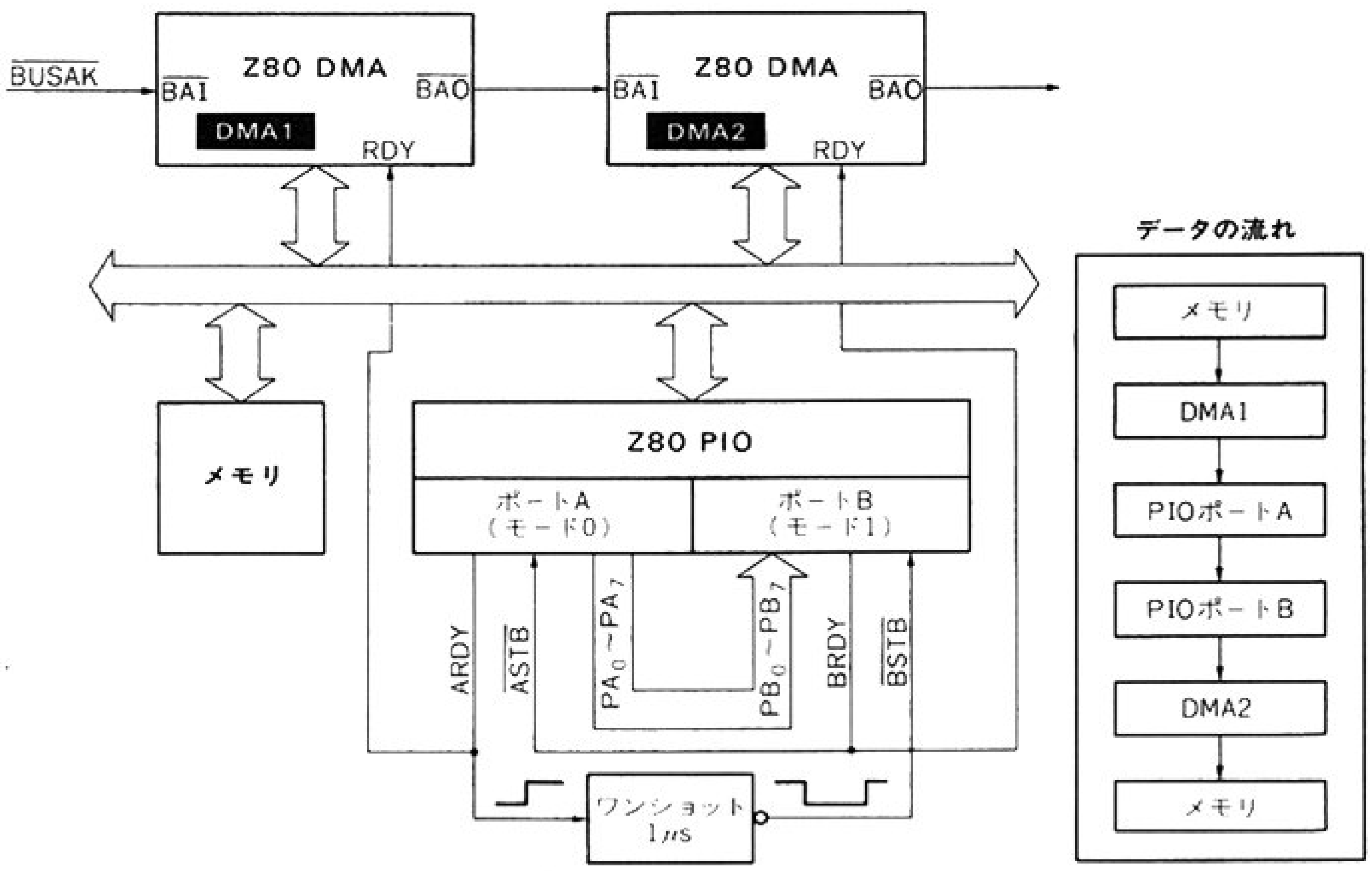
Z80 DMA は Z80 SIO と同様に，使えば使うほど，わからない点，不明な点が出てくる LSI です．自分が意図したとおり動かない場合，それが Z80 DMA のスペックであるのか制限事項であるのか判断に苦しみます．このような場合には，想像力をたくましくして，ああでもない，こうでもないとカット・アンド・トライを繰り返すのが現実です．

〈Prog ⑩〉 I/O → メモリ・データ転送

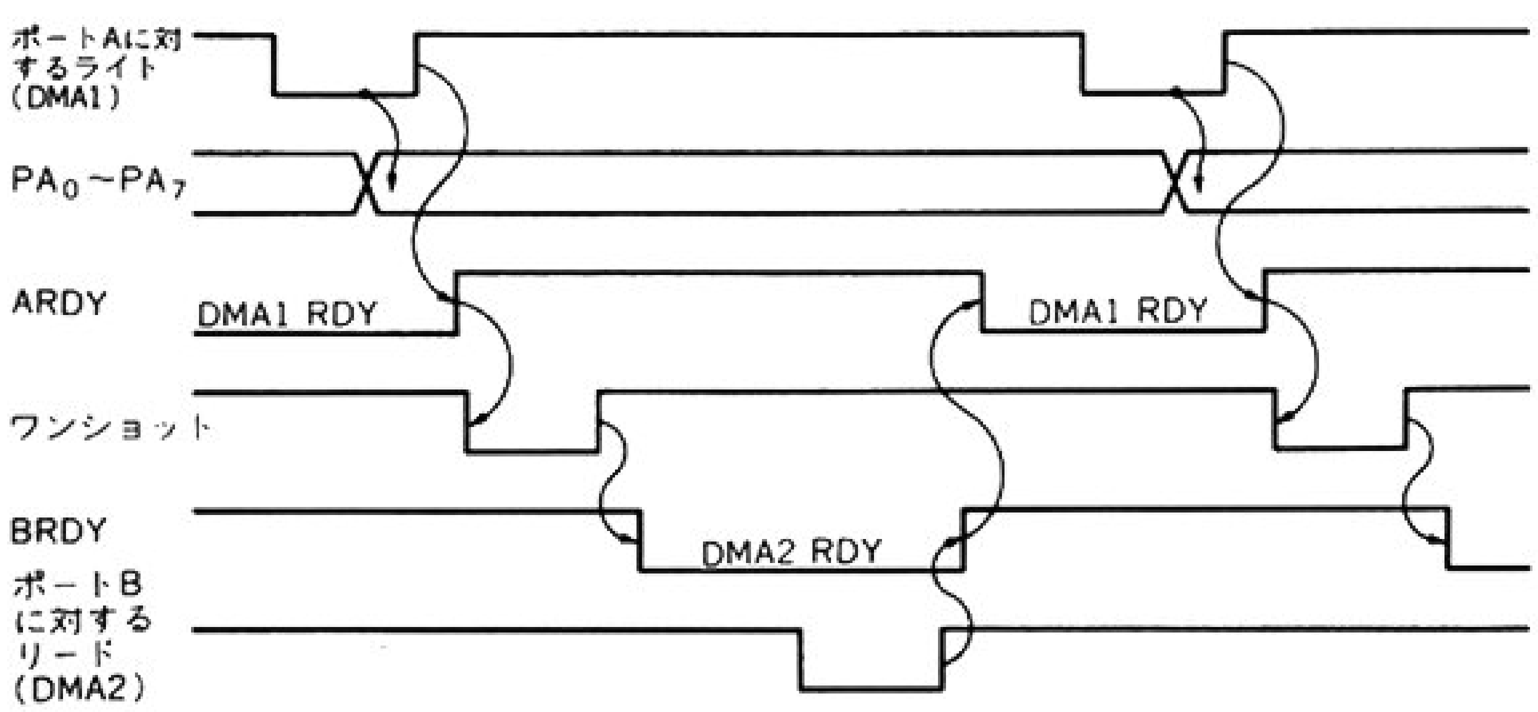
```
                        ;
                        ;       DMA5     I/O TO MEMORY DMA DATA TRANSFER
                        ;       THIS PROGRAM READS DATA FROM PIO PORT B
                        ;       AND WRITE THIS DATA INTO MEMORY VIA DMA.
                        ;
0016                    PIOBD   EQU      16H             ;PIO CHANNEL B DATA
0017                    PIOBC   EQU      17H             ;PIO CHANNEL B CONTROL
                        ;
0018                    DMA     EQU      18H             ;DMA ADDRESS
                        ;
2400                    DEST    EQU      2400H           ;DESTINATION BUFFER
0100                    LENGTH  EQU      256             ;BLOCK LENGTH
0000                            ASEG
                                ORG      2000H
2000    F3              BEGIN:  DI                       ;DISABLE INTERRUPT
2001    31 2138                 LD       SP,STACK        ;SET STACK POINTER
2004    3E 20                   LD       A,20H           ;SET IR
2006    ED 47                   LD       I,A             ;
2008    ED 5E                   IM       2               ;SET MODE 2
                        ;       PIO CHANNEL B INITIALIZATION
200A    3E 4F                   LD       A,4FH           ;MODE 1 (INPUT)
200C    D3 17                   OUT      (PIOBC),A       ;
200E    3E 07                   LD       A,07H           ;DISABLE INTERRUPT
2010    D3 17                   OUT      (PIOBC),A       ;
                        ;       RESET DMA
2012    06 06                   LD       B,6             ;SET COUNT
2014    3E C3                   LD       A,0C3H          ;SET RESET COMMAND
2016    D3 18           LOOP:   OUT      (DMA),A         ;LOAD RESET COMMAND
2018    10 FC                   DJNZ     LOOP            ;LOADED SIX TIMES ?
                                                         ;NO,LOOP
                        ;       INITIALIZATION OF DMA
201A    06 12                   LD       B,DMACL-DMACC   ;LOAD COMMAND LENGTH
201C    0E 18                   LD       C,DMA           ;LOAD DMA ADDRESS
201E    21 203A                 LD       HL,DMACC        ;LOAD COMMAND ADDRESS
2021    ED B3                   OTIR                     ;LOAD COMMAND TO DMA
2023    FB                      EI                       ;ENABLE INTERRUPT
2024    18 FE                   JR       $               ;LOOP ETERNALLY
```

```
                    ;
                    ;       INT ON READY
2026    76          I00:    HALT                          ;SPURIOUS INTERRUPT
                    ;
                    ;       MATCH INTERRUPT
2027    76          I01:    HALT                          ;SPURIOUS INTERRUPT
                    ;
                    ;       END OF BLOCK INTERRUPT
2028    3E 8B       I02:    LD      A,8BH                 ;RE-INITIALIZE STATUS
202A    32 0018             LD      (DMA),A               ;LOAD IT TO DMA
202D    06 04               LD      B,DMACL-DMACCO        ;LOAD COMMAND LENGTH
202F    0E 18               LD      C,DMA                 ;LOAD DMA ADDRESS
2031    21 2048             LD      HL,DMACCO             ;LOAD COMMAND ADDRESS
2034    ED B3               OTIR                          ;LOAD COMMAND TO DMA
2036    FB                  EI                            ;RE-ENABLE INTERRUPT
2037    ED 4D               RETI                          ;
                    ;
                    ;       MATCH,END OF BLOCK INTERRUPT
2039    76          I03:    HALT                          ;SPURIOUS INTERRUPT
                    ;
                    ;       DMA COMMAND CHAIN
                    ;       PORT B IS ASSIGNED TO MEMORY
                    ;       PORT A IS ASSIGNED TO PIO CHANNEL B
203A    C3          DMACC:  DB      0C3H      ;WR6    RESET COMMAND
203B    7D                  DB      7DH       ;WR0    PORT A --> PORT B (TEMP)
                                              ;       PORT A ADRS FOLLOWS
                                              ;       LENGTH FOLLOWS
203C    2400                DW      DEST      ;       DESTINATION ADDRESS
203E    00FF                DW    LENGTH-1    ;       BLOCK LENGTH
2040    14                  DB      14H       ;WR1    PORT A IS MEMORY
                                              ;       ADDRESS INCREMENTS
                                              ;       NO TIMING BYTE FOLLOWS
2041    28                  DB      28H       ;WR2    PORT B IS I/O
                                              ;       ADDRESS FIXED
                                              ;       NO TIMING BYTE FOLLOWS
2042    A0                  DB      0A0H      ;WR3    ENABLE INTERRUPT
2043    95                  DB      95H       ;WR4    PORT B ADRS FOLLOWS
                                              ;       INT CTL BYTE FOLLOWS
                                              ;       BYTE MODE TRANSFER
2044    16                  DB      PIOBD     ;       PORT B ADDRESS
2045    32                  DB      32H       ;       INT CTL BYTE
                                              ;       INT AT END OF BLOCK
                                              ;       STATUS AFFECTS VECTOR
                                              ;       VECTOR FOLLOWS
2046    F0                  DB DBINTV-BEGIN   ;       VECTOR
2047    82                  DB      82H       ;WR5    READY ACTIVE LOW
                                              ;       CE/ ONLY
2048    CF          DMACCO: DB      0CFH      ;WR6    LOAD ADDRESS TO PORT A
2049    01                  DB      01H       ;WR0    PORT B --> PORT A
204A    CF                  DB      0CFH      ;WR6    LOAD ADDRESS TO PORT B
204B    87                  DB      87H       ;WR6    ENABLE DMA
204C                DMACL   EQU     $
                    ;
                    ;
                            ORG     BEGIN+0F0H
20F0    2026        DBINTV: DW      I00                   ;INT ON READY
20F2    2027                DW      I01                   ;MATCH INTERRUPT
20F4    2028                DW      I02                   ;END OF BLOCK INT
20F6    2039                DW      I03                   ;MATCH,END OF BLOCK
                    ;
                    ;
20F8                        DS      64                    ;STACK AREA
2138                STACK   EQU     $
                    ;
                    ;
                            END
```

〈図83〉
2個のDMAによるPIOのループ・バック



〈図84〉
図83の構成が動作するタイミング



こうした事柄についても言及するとよかったのですが，メーカーの種類，バージョン等についてすべてのテストを行わずに，安易な判断を下すようなことはできませんので，今回は断念しました．

Z80 DMAのような複雑なLSIを使う場合には，直接アプリケーションに取り組まず，まず簡単な評価用ハードウェアを作って，動作を確認してみる必要があります．それにより不明な点をできるだけなくしてから，本物のアプリケーションに適用していくのが賢明な方法です．

本項で示したプログラム例は，本書の執筆に当たって製作したシングル・ボード・コンピュータ上で実際に動作させ，目標どおり動くことを確認したプログラムです．

筆者が始めてZ80 DMAを使ったのは5年程前のことですが，当時はアプリケーション・マニュアルもなく，もちろん雑誌の記事等もありませんでした．したがって，闇夜でカラスを探すようなことをしながら，Z80 DMAを動かした記憶があります．本項がZ80 DMAを使おうとしている読者のための一助となれば幸いです．

〈Prog ⓫〉 二つのDMAによるPIOループ・バック

```
                          ;
                          ;        DMA3     Z80 DMA'S AND Z80 PIO LOOP BACK TEST
                          ;
                          ;        DMA1 READS DATA FROM MEMORY AND WRITES IT TO PIO
                          ;        CHANNEL A. THIS DATA IS TRANSFERED TO PIO
                          ;        CHANNEL B BY ARDY VIA ONE SHOT.
                          ;        DMA2 READS THE DATA FROM PIO CHANNEL B AND
                          ;        WRITES IT INTO MEMORY.
                          ;
0014                      PIOAD    EQU      14H              ;PIO CHANNEL A DATA
0015                      PIOAC    EQU      15H              ;PIO CHANNEL A CONTROL
0016                      PIOBD    EQU      16H              ;PIO CHANNEL B DATA
0017                      PIOBC    EQU      17H              ;PIO CHANNEL B CONTROL
                          ;
0018                      DMA1     EQU      18H              ;DMA1 ADDRESS
0010                      DMA2     EQU      10H              ;DMA2 ADDRESS
                          ;
2200                      SRCE     EQU      2200H            ;SOURCE ADDRESS
2300                      DEST     EQU      2300H            ;DESTINATION ADDRESS
00FF                      LENGTH   EQU      256-1            ;TRANSFER LENGTH (256)
                          ;
                          ;
0000                               ASEG
                                   ORG      2000H
2000      F3              BEGIN:   DI                        ;DISABLE INTERRUPT
2001      31 2140                  LD       SP,STACK         ;SET STACK POINTER
2004      3E 20                    LD       A,20H            ;SET IR
2006      ED 47                    LD       I,A              ;
2008      ED 5E                    IM       2                ;MODE 2 INTERRUPT
                          ;
                          ;        SOURCE DATA INITIALIZATION
200A      01 0000                  LD       BC,0             ;SET DATA AND COUNT
200D      21 2200                  LD       HL,SRCE          ;LOAD SRCE ADDRESS
2010      71              BEGINO:  LD       (HL),C           ;SET DATA INTO SOURCE
2011      23                       INC      HL               ;BUMP SOURCE POINTER
2012      0C                       INC      C                ;UPDATE DATA
2013      10 FB                    DJNZ     BEGINO           ;FILLED 256 BYTES ?
                                                             ;NO,BEGINO
                          ;
                          ;        PIO CHANNEL A INITIALIZATION
2015      3E 0F                    LD       A,0FH            ;SET MODE 0 (OUTPUT)
2017      D3 15                    OUT      (PIOAC),A        ;
2019      3E 07                    LD       A,07H            ;DISABLE INTERRUPT
201B      D3 15                    OUT      (PIOAC),A        ;
                          ;
                          ;        PIO CHANNEL B INITIALIZATION
201D      3E 4F                    LD       A,4FH            ;SET MODE 1 (INPUT)
201F      D3 17                    OUT      (PIOBC),A        ;
2021      3E 07                    LD       A,07H            ;DISABLE INTERRUPT
2023      D3 17                    OUT      (PIOBC),A        ;
                          ;
                          ;        RESET DMA1
2025      3E C3                    LD       A,0C3H           ;LOAD RESET COMMAND
2027      06 06                    LD       B,6              ;SET COUNT
2029      D3 18           LOOP:    OUT      (DMA1),A         ;RESET COMMAND TO DMA1
202B      10 FC                    DJNZ     LOOP             ;LOADED SIX TIMES ?
                                                             ;NO,LOOP
                          ;
                          ;        DMA1 INITIALIZATION
202D      06 11                    LD       B,DMA1CL-DMA1CC  ;DMA1 COMMAND LENGTH
202F      0E 18                    LD       C,DMA1           ;LOAD DMA1 ADDRESS
2031      21 20A6                  LD       HL,DMA1CC        ;DMA1 COMMAND ADDRESS
2034      ED B3                    OTIR                      ;LOAD COMMAND TO DMA1
                          ;
```

```
                          ;       RESET DMA2
2036    3E C3                     LD      A,0C3H             ;LOAD RESET COMMAND
2038    06 06                     LD      B,6                ;SET COUNT
203A    D3 10             LOOP0:  OUT     (DMA2),A           ;RESET COMMAND TO DMA2
203C    10 FC                     DJNZ    LOOP0              ;LOADED SIX TIMES
                                                             ;NO,LOOP0
                          ;
                          ;       DMA2 INITIALIZATION
                          ;       NOTE THAT HL POINTS DMA2CC NOW.
203E    06 11                     LD      B,DMA2CL-DMA2CC    ;DMA2 COMMAND LENGTH
2040    0E 10                     LD      C,DMA2             ;LOAD DMA2 ADDRESS
2042    ED B3                     OTIR                       ;LOAD COMMAND TO DMA2
                          ;
                          ;       NOW ENABLE DMA'S
2044    3E 87                     LD      A,87H              ;ENABLE DMA
2046    D3 18                     OUT     (DMA1),A           ;ENABLE DMA1
2048    D3 10                     OUT     (DMA2),A           ;ENABLE DMA2
204A    06 00                     LD      B,0                ;CLEAR DMA1 INT FLAG
                          ;       B=DMA1 INTERRUPT OCCURRED FLAG
204C    FB                        EI                         ;ENABLE INTERRUPT
204D    18 FE                     JR      $                  ;LOOP ETERNALLY
                          ;
                          ;
                          ;       DMA1 INT ON RDY INTERRUPT
204F    76                I100:   HALT                       ;SPURIOUS INTERRUPT
                          ;
                          ;       DMA1 MATCH INTERRUPT
2050    76                I101:   HALT                       ;SPURIOUS INTERRUPT
                          ;
                          ;       DMA1 END OF BLOCK INTERRUPT
2051    06 01             I110:   LD      B,1                ;SET DMA1 INT OCCURRED
2053    3E 8B                     LD      A,8BH              ;CLEAR END OF BLOCK INT
2055    D3 18                     OUT     (DMA1),A           ;
2057    FB                        EI                         ;ENABLE INTERRUPT
2058    ED 4D                     RETI                       ;
                          ;
                          ;       DMA1 MATCH/END OF BLOCK INTERRUPT
205A    76                I111:   HALT                       ;SPURIOUS INTERRUPT
                          ;
                          ;
                          ;       DMA2 INT ON RDY INTERRUPT
205B    76                I200:   HALT                       ;SPURIOUS INTERRUPT
                          ;
                          ;       DMA2 MATCH INTERRUPT
205C    76                I201:   HALT                       ;SPURIOUS INTERRUPT
                          ;
                          ;       DMA2 END OF BLOCK INTERRUPT
205D    3E 8B             I210:   LD      A,8BH              ;CLEAR END OF BLOCK INT
205F    D3 10                     OUT     (DMA2),A           ;
2061    78                        LD      A,B                ;DMA1 INT OCCURRED ?
2062    B7                        OR      A                  ;
2063    20 01                     JR      NZ,I210A           ;YES,I210A
2065    76                        HALT                       ;ERROR STOP
                          ;
                          ;       NOW VERIFY LOOP BACKED DATA
2066    11 2200           I210A:  LD      DE,SRCE            ;LOAD SOURCE ADDRESS
2069    21 2300                   LD      HL,DEST            ;LOAD DESTINATION ADRS
206C    06 00                     LD      B,0                ;SET COUNTER (256)
206E    1A                I210B:  LD      A,(DE)             ;LOAD SOURCE DATA
206F    BE                        CP      (HL)               ;DATA MATCH ?
2070    28 01                     JR      Z,I201C            ;YES,I210C
2072    76                        HALT                       ;DATA ERROR STOP
2073    13                I201C:  INC     DE                 ;BUMP SOURCE POINTER
2074    23                        INC     HL                 ;BUMP DESTINATION PTR
2075    10 F7                     DJNZ    I210B              ;CHECKED ALL ?
                                                             ;NO,I210B
```

```
                                ;
                                ;         ROTATE  SOURCE DATA
2077     3A 2200                          LD      A,(SRCE)          ;SAVE TOP OF DATA
207A     11 2200                          LD      DE,SRCE           ;LOAD SOURCE ADDRESS
207D     21 2201                          LD      HL,SRCE+1         ;LOAD "FROM" ADDRESS
2080     01 00FF                          LD      BC,LENGTH         ;LOAD LENGTH TO MOVE
2083     ED B0                            LDIR                      ;ROTATE DATA TO LEFT
2085     12                               LD      (DE),A            ;PUT TOP OF DATA INTO
                                                                    ;TAIL
                                ;
                                ;         RELOAD COUNT AND ADDRESS TO CYCLE TEST
2086     06 04                            LD      B,RELDML-RELDMA   ;LOAD COMMAND LENGTH
2088     0E 18                            LD      C,DMA1            ;LOAD DMA1 ADDRESS
208A     21 20A1                          LD      HL,RELDMA         ;LOAD COMMAND ADDRESS
208D     ED B3                            OTIR                      ;LOAD COMMAND TO DMA1
208F     06 04                            LD      B,RELDML-RELDMA   ;LOAD COMMAND LENGTH
2091     0E 10                            LD      C,DMA2            ;LOAD DMA2 ADDRESS
2093     21 20A1                          LD      HL,RELDMA         ;LOAD COMMAND ADDRESS
2096     ED B3                            OTIR                      ;LOAD COMMAND TO DMA2
                                ;         NOTE THAT INTERRUPT FLAG (REG. B) IS CLEARED.
2098     3E 87                            LD      A,87H             ;RE-ENABLE DMA
209A     D3 18                            OUT     (DMA1),A          ;RE-ENABLE DMA1
209C     D3 10                            OUT     (DMA2),A          ;RE-ENABLE DMA2
209E     FB                               EI                        ;ENABLE INTERRUPT
209F     ED 4D                            RETI                      ;
                                ;
                                ;         COMMAND CHAIN TO RE-ENABLE DMA'S
20A1     05                     RELDMA:   DB      05H      ;WR0     PORT A --> PORT B (TEMP)
20A2     CF                               DB      0CFH     ;WR6     LOAD COUNT & ADRS TO A
20A3     01                               DB      01H      ;WR0     PORT B --> PORT A
20A4     CF                               DB      0CFH     ;WR6     LOAD COUNT & ADRS TO B
20A5                            RELDML    EQU     $                 ;
                                ;
                                ;
                                ;         DMA2 MATCH, END OF BLOCK INTERRUPT
20A5     76                     I211:     HALT                      ;SPURIOUS INTERRUPT
                                ;
                                ;
                                ;         DMA1 COMMAND CHAIN
                                ;         PORT A IS ASSIGNED TO PIO CHANNEL A
                                ;         PORT B IS ASSIGNED TO MEMORY
20A6     C3                     DMA1CC:   DB      0C3H     ;WR6     RESET DMA1
20A7     6D                               DB      6DH      ;WR0     TRANSFER
                                                           ;        PORT A --> PORT B (TEMP)
                                                           ;        ADDRESS FOLLOWS (LOWER)
                                                           ;        LENGTH FOLLOWS
20A8     14                               DB      PIOAD    ;        PORT A DESTINATION
20A9     00FF                             DW      LENGTH   ;        PORT A LENGTH
20AB     2C                               DB      2CH      ;WR1     PORT A IS I/O
                                                           ;        ADDRESS FIXED
                                                           ;        NO TIMING BYTE FOLLOWS
20AC     10                               DB      10H      ;WR2     PORT B IS MEMORY
                                                           ;        ADRESS INCREMENTS
                                                           ;        NO TIMING BYTE FOLLOWS
20AD     A0                               DB      0A0H     ;WR3     ENABLE INTERRUPT
20AE     9D                               DB      9DH      ;WR4     PORT A ADRS FOLLOWS
                                                           ;        INTERRUPT CONTROL BYTE
                                                           ;        FOLLOWS
                                                           ;        BYTE MODE TRANSFER
20AF     2200                             DW      SRCE     ;        PORT B ADDRESS
20B1     32                               DB      32H      ;        INTERRUPT CONTROL BYTE
                                                           ;        INT AT END OF BLOCK
                                                           ;        STATUS AFFECTS VECTOR
                                                           ;        VECTOR FOLLOWS
20B2     F0                               DB DAINTV-BEGIN  ;        INTERRUPT VECTOR
20B3     82                               DB      82H      ;WR5     READY AVTIVE LOW
                                                           ;        CE/ ONLY
                                                           ;        STOP AT END OF BLOCK
```

```
20B4    CF              DB      0CFH    ;WR6    LOAD COUNT & ADRS TO A
20B5    01              DB      01H     ;WR0    PORT B --> PORT A
20B6    CF              DB      0CFH    ;WR6    LOAD COUNT & ADRS TO B
20B7            DMA1CL  EQU     $               ;
                ;
                ;
                ;       DMA2 COMMAND CHAIN
                ;       PORT A IS ASSIGNED TO MEMORY
                ;       PORT B IS ASSIGNED TO PIO CHANNEL B
20B7    C3      DMA2CC: DB      0C3H    ;WR6    RESET DMA2
20B8    7D              DB      7DH     ;WR0    PORT A --> PORT B (TEMP)
                                        ;       PORT A ADRS FOLLOWS
                                        ;       LENGTH FOLLOWS
20B9    2300            DW      DEST    ;       DESTINATION ADDRESS
20BB    00FF            DW      LENGTH  ;       LENGTH
20BD    14              DB      14H     ;WR1    PORT A IS MEMORY
                                        ;       ADDRESS INCREMENTS
20BE    28              DB      28H     ;WR2    PORT B IS I/O
                                        ;       ADDRESS FIXED
                                        ;       NO TIMING BYTE FOLLOWS
20BF    A0              DB      0A0H    ;WR3    ENABLE INTERUPT
20C0    95              DB      95H     ;WR4    PORT B ADRS FOLLOWS
                                        ;       INTERRUPT CONTROL BYTE
                                        ;       FOLLOWS
                                        ;       BYTE MODE TRANSFER
20C1    16              DB      PIOBD   ;       PORT B ADDRESS
20C2    32              DB      32H     ;       INTERRUPT CONTROL BYTE
                                        ;       INT AT END OF BLOCK
                                        ;       STATUS AFFECTS VECTOR
                                        ;       VECTOR FOLLOWS
20C3    F8              DB DBINTV-BEGIN ;       INTERRUPT VECTOR
20C4    82              DB      82H     ;WR5    READY ACTIVE LOW
                                        ;       CE/ ONLY
20C5    CF              DB      0CFH    ;WR6    LOAD COUNT & ADRS TO A
20C6    01              DB      01H     ;WR0    PORT B --> PORT A
20C7    CF              DB      0CFH    ;WR6    LOAD COUNT & ADRS TO B
20C8            DMA2CL  EQU     $               ;
                ;
                ;
                        ORG     BEGIN+0F0H
                ;       DMA1 INTERRUPT VECTOR
20F0    204F    DAINTV: DW      I100            ;DMA1 INT ON RDY
20F2    2050            DW      I101            ;DMA1 MATCH
20F4    2051            DW      I110            ;DMA1 END OF BLOCK
20F6    205A            DW      I111            ;DMA1 MATCH, END OF BLK
                ;
                ;       DMA2 INTERRUPT VECTOR
20F8    205B    DBINTV: DW      I200            ;DMA2 INT ON RDY
20FA    205C            DW      I201            ;DMA2 MATCH
20FC    205D            DW      I210            ;DMA2 END OF BLOCK
20FE    20A5            DW      I211            ;DMA2 MATCH, END OF BLK
                ;
                ;
2100                    DS      64              ;STACK AREA
2140            STACK   EQU     $               ;
                ;
                        END
```

3.4 Z80 SIO編

▶Z80 Serial I/O◀

Z80 SIO (Serial Input/Output) は Z80 ファミリのコミュニケーション・コントローラであり，非同期通信，同期通信はもとより，SDLC の制御までも可能な，マルチ・プロトコル・コミュニケーション・コントローラです．

本稿ではこの Z80 SIO の機能，インターフェース，プログラミングについて詳細に説明をしますが，その説明の前段階として「シリアル・コミュニケーションのあらまし」と称して，Z80 SIO が取り扱うことのできる三つの通信方式について説明をすることから始めます．

シリアル・コミュニケーションのあらまし

データをある地点から他の地点に伝送する（コミュニケーション）方法として，もっとも単純かつ明解な方法はパラレル (Parallel――並列) 伝送方式です．図1にパラレル伝送方式の概念図を示します．図1の例では8ビットのデータをA地点からB地点に伝送するのにデータ・ラインが8本要ることになります．

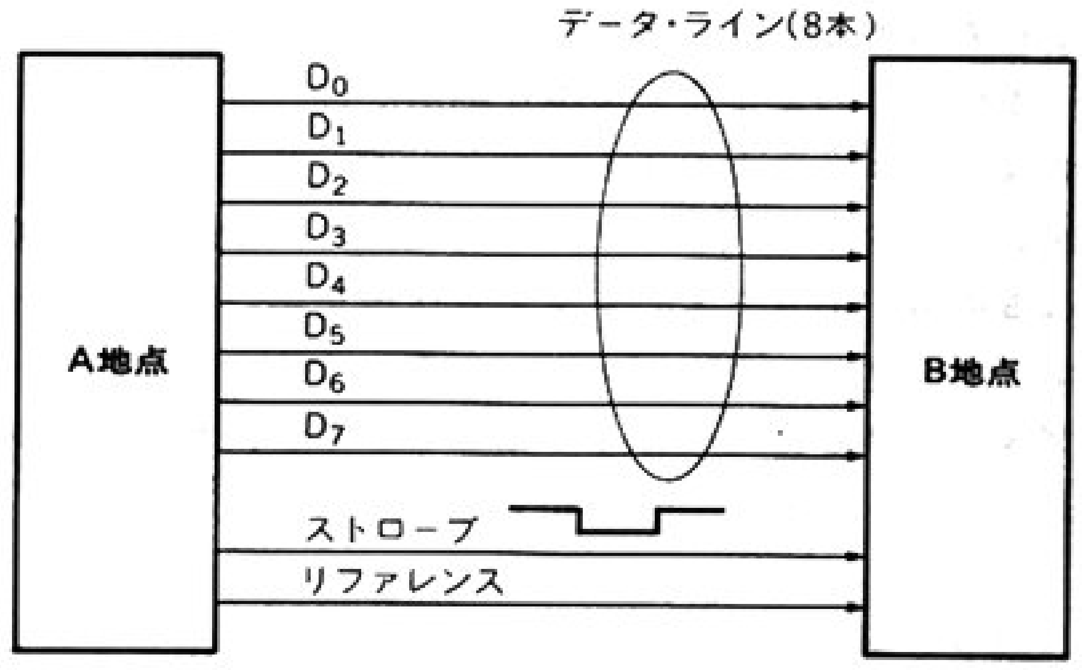
A地点とB地点間の距離が，それほど長くないときには，8本のデータ・ラインをA，B両地点間にしいたとしても，電線のコストはがまんできる範囲内にありますが，距離が長い場合にはばく大なものになり，電線の数を少なくするために別の方法を採り入れる必要があります．その方法がシリアル (Serial――直列) 伝送方式です．

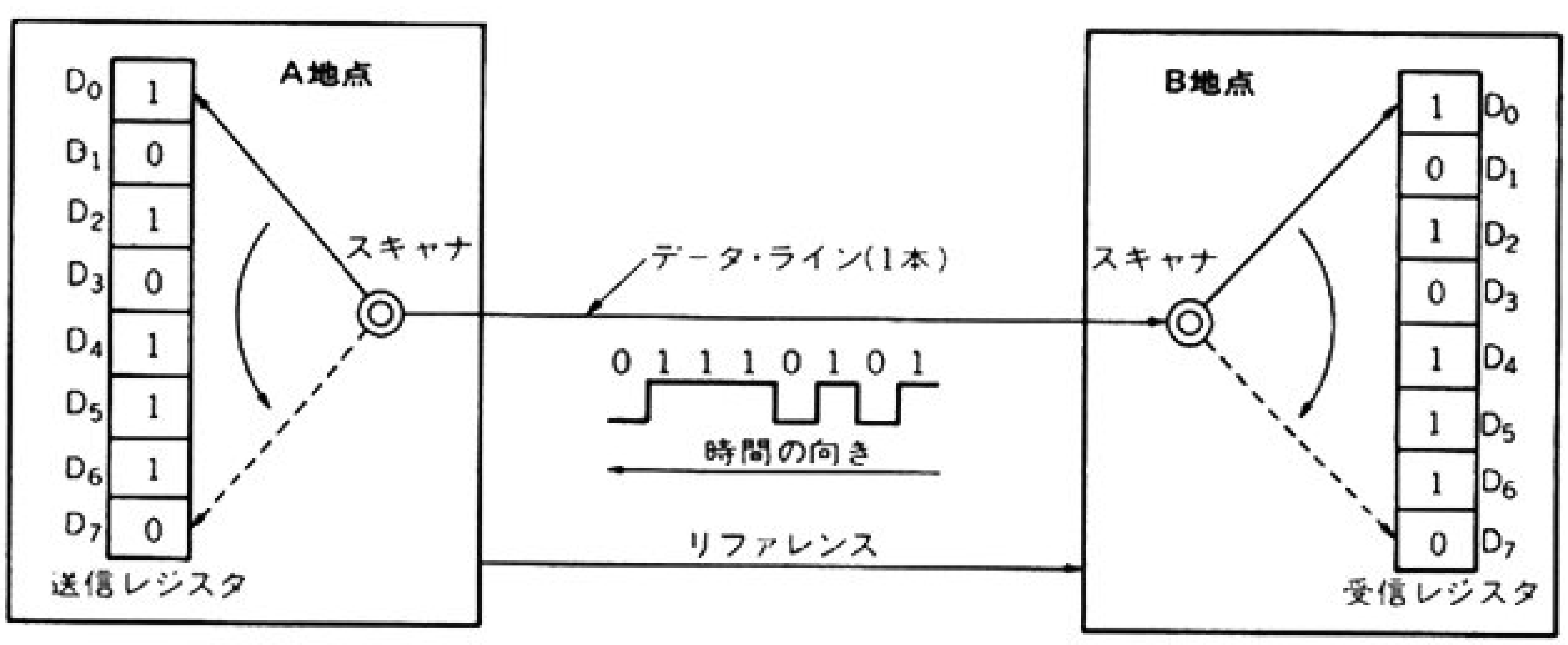
図2にシリアル伝送方式の概念図を示します．この方式では8ビットのデータをA地点からB地点に伝送するのに1本のデータ・ラインですみます．

A地点では8ビットのデータをスキャナにより，一定時間間隔で1ビットずつ走査し，データ・ビットの状態（1か0）をデータ・ライン上に乗せます．B地点のスキャナはA地点のスキャナと同一時間間隔で，データ・ライン上の状態をサンプリングし，順次その状態を受信レジスタにセットします．

このようにシリアル伝送方式では，送信側でデータ

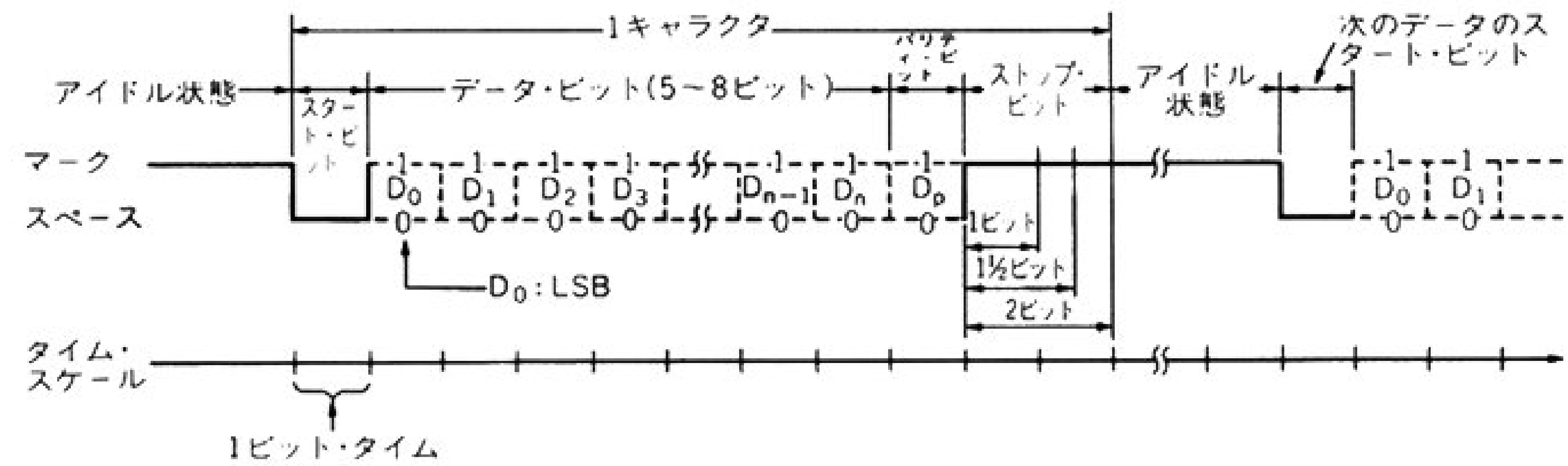
〈図1〉 パラレル伝送方式の概念．8ビットのデータを伝送するのに8本のデータ・ライン＋α本が必要





〈図2〉
シリアル伝送方式の概念図

〈図3〉
非同期通信方式のデータ・フォーマット



を並列 → 直列変換をし，受信側ではデータ・ライン上の状態を直列 → 並列変換してデータを復元します．

以上の話からわかるように，パラレル伝送方式ではデータを空間分割して伝送しており，シリアル伝送方式では，データを時分割して伝送しているということができます．

シリアル伝送方式には，データ・ラインが1本ですむという大きなメリットがありますが，時分割方式であるため，単位時間当たりに伝送できるデータ量はパラレル伝送方式に比べて，データのビット数分の1になります．つまり図2の例のように8ビット・データを伝送する場合には，8分の1ということになります．また送信側で直列 → 並列変換をし，受信側で並列 → 直列変換をしなければならないので，必然的に回路構成は複雑になります．

とはいえ，現在華やかなオンライン・リアルタイム・システムやTSS等はすべてこのシリアル・コミュニケーションで行われており，今後ますます発展していくであろうことは，誰れしもが認めるところです．

また，このような大規模な応用だけではなく，最近ではパソコンやターミナルのキーボードやプリンタのインターフェースとして，このシリアル伝送が用いられる風潮があります．シリアル伝送を用いれば，キーボードやプリンタとパソコンやターミナルを結ぶケーブルの太さを極端に細くすることができ，かつケーブルのコストも低減できるからです．

以上でシリアル・コミュニケーションの概念をおわかりいただけたことと思いますが，送信側と受信側の同期を図るため，シリアル伝送の方式はもう少し複雑です．送信側と受信側の同期を図るための方法として，非同期通信方式と同期通信方式があります．同期をとるための方法としての非同期通信方式というのは，言語上あきらかな矛盾がありますので，調歩同期式という用語もありますが，一般的には非同期通信方式と呼んでいますので，本稿でもそれで通すことにします．

非同期通信方式 (Asynchronous Communication)

非同期通信方式では，データの一単位（たとえば8ビット）の始めと終わりを示すために，スタート・ビットおよびストップ・ビットという概念が用いられます．図3に非同期通信方式におけるデータ・フォーマットを示します．

この方式では，データ・ライン上にデータが存在していないときには，データ・ラインの状態はマーク状態（"1"）となっており，一単位のデータが始まる前には必ず，1ビット・タイム（1ビットを伝送するのに要する時間）期間，データ・ラインの状態はスペース状態（"0"）になります．

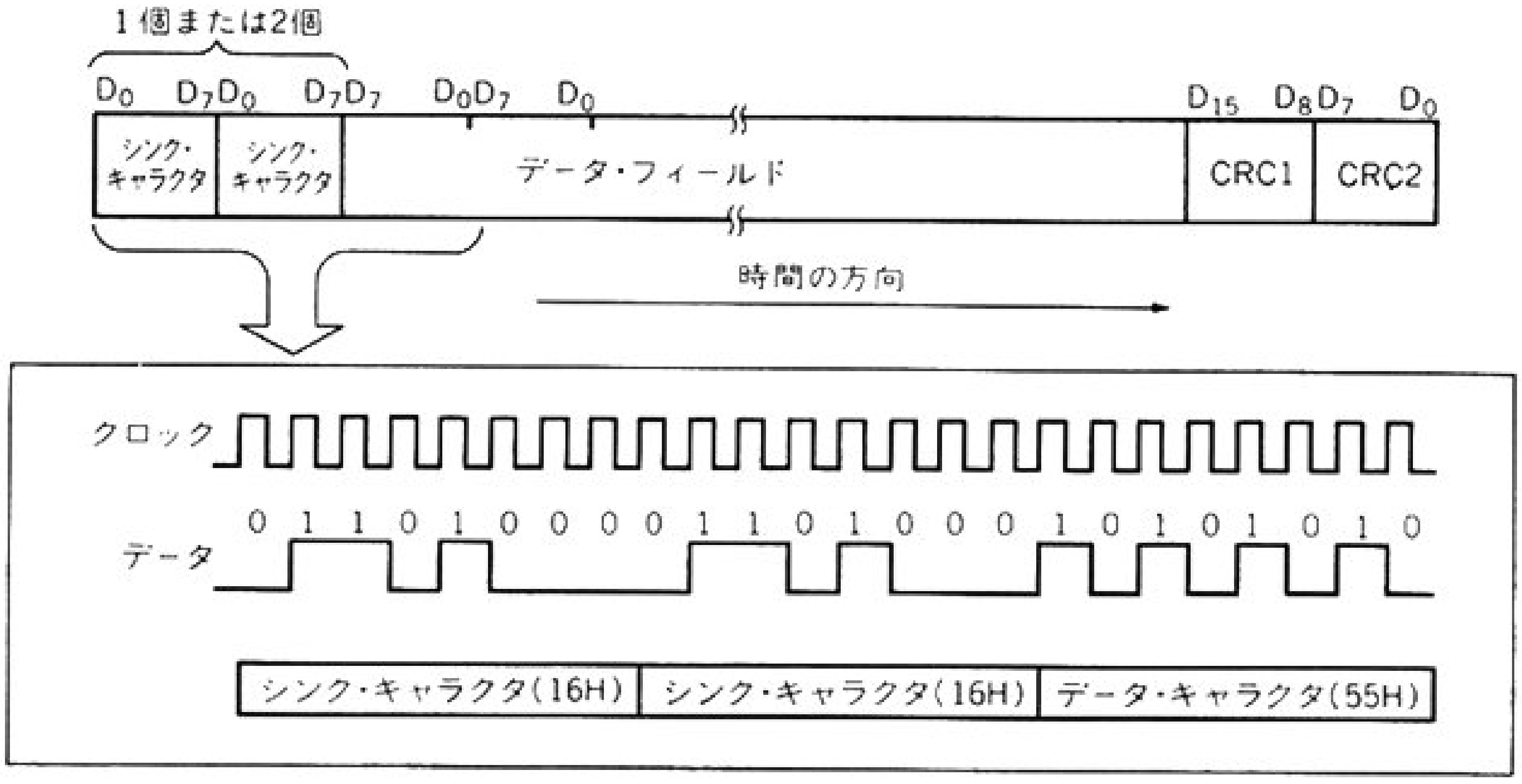
この1ビット・タイムのスペース状態のことをスタート・ビットと呼びます．

このスタート・ビットの後にデータ・ビット（図3では8ビット）が続きます．データ・ビットの後にはストップ・ビットという，1～2ビット・タイムのマーク状態が続きます．図3ではストップ・ビットが1ビット・タイムの場合を示してあります．

したがって一単位のデータは，スタート・ビット，データ・ビット，それにストップ・ビットから構成されます．

この方式では8ビット・データを1ストップ・ビット・タイムで伝送するのに要する時間は10ビット・タイムです．したがって，データ・ライン上を流れる信号の2割は，データの内容とはまったく関係のないものであり，データ・ラインの使用効率は高いとはいえません．

ただし，データ・ライン上の信号により，送信側と受信側の同期をはかりますので，同期用の信号線が不要であるというメリットがあります．この方式は，スタート・ビットとストップ・ビットにより，データ・ラインの同期をはかりますので，スタート／ストップ方式と呼ばれることもあります．



〈図4〉
同期通信方式のデータ・フォーマット

同期通信方式 (Synchronous Communication)

同期通信方式は，非同期通信方式におけるデータ・ラインの使用効率の低さをカバーするために考え出された方式です．図4に同期通信方式における，データ・フォーマットを示します．

この方式では，送信側またはモデムから受信クロックを受信側に送出し，受信側ではこのクロックに同期して，データ・ラインの状態をサンプリングします．

とはいえ，この方式においてもデータ・ライン上を直列に流れてくる信号中に，データの開始時点を示すものがなくてはなりません．同期通信方式ではデータ・ブロックの開始を示すために，シンク・キャラクタ(Synch. Character) が用いられます．送信側では，データ・ブロックの先頭に1個以上のシンク・キャラクタを挿入して，データを送信します．

受信側では受信クロックに同期して，データ・ラインの状態をサンプリングし，1個以上のシンク・キャラクタを検出すると，データ・ブロックの開始であると認識し，ブロック・データの受信を行います．

通常は同期確立のため，1または2個のシンク・キャラクタが用いられますが，1個の場合をシングル・シンク・モード，2個の場合をダブル・シンク・モードと呼びます．

1または2個のシンク・キャラクタを検出した受信側は，それ以降のデータ・ラインの状態をデータ・ブロックの内容であると思いこみますが，データ・ブロック中に検出されたシンク・キャラクタはアイドル・シンク (Idle Sync)・キャラクタとして無視します．

このようにデータ・ブロック中にシンク・キャラクタが検出されるような状態は，送信側がデータ・ブロックの開始時に3個以上のシンク・キャラクタを挿入する場合や，送信側の都合で連続的にデータを送信できなかった場合に生じます．

同期通信方式では，データを送出すべき時間に，送信データの用意ができなかったような場合には，送信側はアイドル・シンク・キャラクタをデータ・ライン上に乗せます．

非同期通信方式では，送信側に送るべきデータがないときには，データ・ラインの状態をマーク状態にしておけばすみますが，この同期通信方式では1ブロックのデータを送信し終わるまで，データ・ライン上の同期を持続させておく必要があります．

この方式ではデータ・ブロックの終了を特殊キャラクタ (Special Character) により行います．この特殊キャラクタとしては通常 ETX または ETB が用いられます．この ETX の後には伝送されたブロック・データの正当性を保証するために，CRC(Cyclic Redandancy Check Character) または BCC (Block Check Character) が続き，データ・ブロックは終了します．

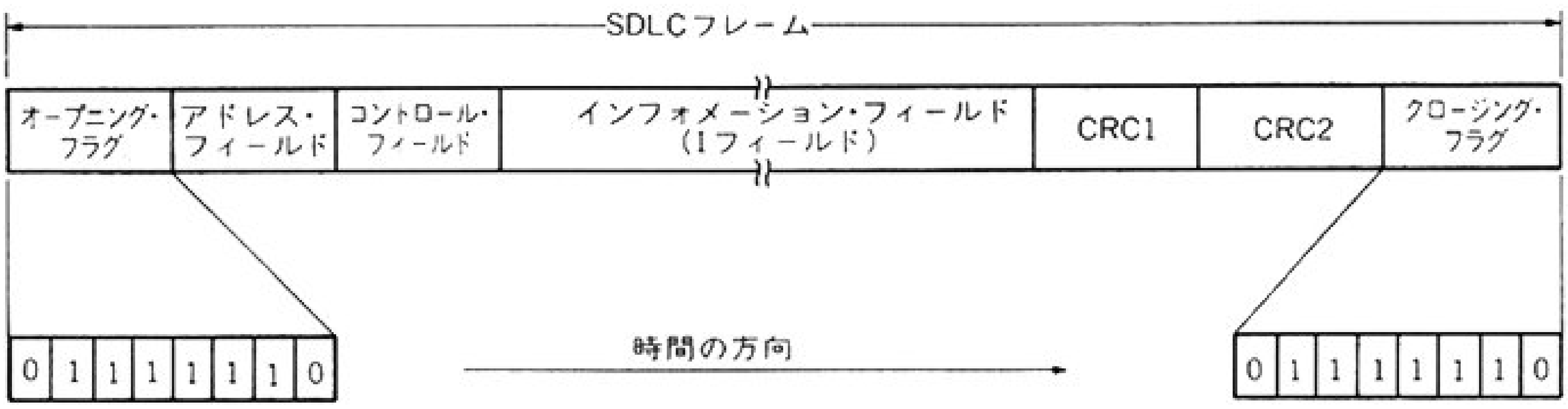
同期通信方式では，データ・ラインの同期に用いられるのはシンク・キャラクタだけですので，データ・ブロックが長い場合にはデータ・ライン上を流れる信号のほとんど100%がデータであり，データ・ラインの使用効率は非常に高いということができます．

SDLC方式 (Synchronous Data Link Control)

このSDLC方式も同期通信方式ですが，データ・ブロック (フレームと呼ぶ) がフラグと呼ばれる特殊キャラクタにより挟まれており，フレーム中にはどのようなビット・パターンが含まれていてもかまいません．図5にSDLC方式におけるフレーム・フォーマットを示します．

フレームの開始を示すフラグのことを，オープニング・フラグ (Openning Flag) と呼び，フレームの終了を示すフラグのことを，クロージング・フラグ (Clos-

〈図5〉
SDLCのフォーマット



ing Flag) と呼びます．これらのフラグのビット・パターンは "01111110" ($7E_{16}$) です．

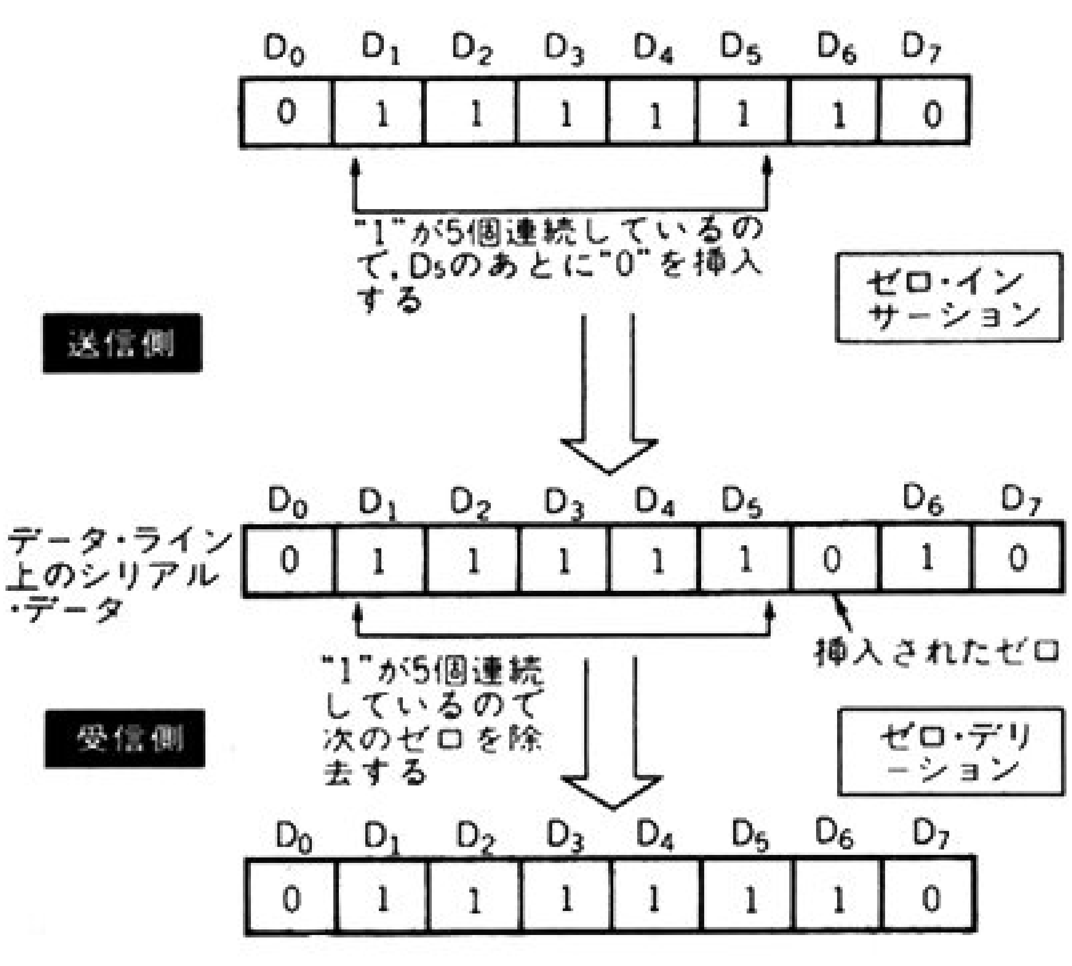
オープニング・フラグの後には，8ビットのアドレス・フィールド（従局アドレス），コントロール・フィールドが続き，その後にインフォメーション・フィールド（Iフィールド），および CRC フィールドが続きます．

SDLCではフラグ間に挟まれているフィールドのデータ・パターンがフラグのビット・パターンと決して一致しないようなくふうがなされています．このくふうのことを，ゼロ・インサーション (Zero Insertion) およびゼロ・デリーション (Zero Deletion) と呼びます．

SDLC の送信側では，アドレス・フィールドから CRC フィールドまでのデータの中に，連続した5個の "1" を検出すると，自動的にゼロを一つ挿入します．したがって，ビット・パターン"01111110"は"011111010"に変換されて送信されます．この過程をゼロ・インサーションといいます．

これに対して受信側では，連続した5個の "1" を検出すると，その次の "0" を無視し，データ中には含めません．この過程をゼロ・デリーションと呼びます．したがって，SDLC では任意のビット・パターンを含むフレームを任意の長さで送信することができます．図6にゼロ・インサーションおよびゼロ・デリーションの例を示します．

〈図6〉 ゼロ・インサーション／ゼロ・デリーション



受信の開始は受信側がオープニング・フラグを検出することにより開始されます．受信側はオープニング・フラグを検出すると，次のデータをアドレス・フィールドであると解釈し，アドレス・フィールドの内容が自分のステーション・アドレスと一致するか否かをチェックします．

一致した場合には，それ以降のデータをデータ・フィールドと解釈し，受信したデータを内部に取り込みます．1フレームの受信は，オープニング・フラグと同じビット・パターンのクロージング・フラグを検出することにより終了します．

アドレス・フィールドはそのフレームを送信すべきステーションのアドレス（従局アドレス）を示しますが，このフィールドの内容が FF_{16} である場合には，ブロードキャスト・アドレス (Broadcast Address) となり，当該フレームがすべてのステーションに対するものであることを示します．

また 00_{16}の値を持つアドレスはノー・ステーション・アドレス (No Station Address) としてリザーブされており，このアドレスを持ったフレームに対しては，どのステーションも応答してはいけないことになっています．

また SDLC にはアボート (Abort) という機能があり，送信側が何らかの理由でフレームの送信を途中で打ち切ることができます．この場合送信側は最低7個の連続した "1" をゼロ・インサーションせずに送信します．これを受信すると受信側はフレームが途中でアボートされたものと解釈し，その時点に受信していたフレームをキャンセルします．

前述の同期通信方式ではデータ・ブロック内に種々の特殊キャラクタを用いて伝送制御を行いますので，伝送制御とは関係ないようなユーザ・データ中に存在していてはいけないようなコードがいくつかありますが，この SDLC ではデータ・フィールド中のコントロール・フィールドにより伝送制御が行われますので，インフォメーション・フィールドには任意のビット・

パターンを含ませることができます．

このような意味で，SDLCのことをビット・オリエンテッド・プロトコル（BOP）と呼び，これに対してふつうの同期通信方式のことをキャラクタ・オリエンテッド・プロトコル（COP）と呼びます．

Z80 SIO
(Serial I/O)

Z80 SIOの機能

Z80 SIOは前項で述べた三つの通信方式を制御できるコミュニケーション・コントローラです．発売当時は値段が高過ぎて，とても素人に買えるようなシロモノでもなかったし，SIOのマニュアルを見ても何となく複雑で，ちょっとやそっとでは動かせる自信もありませんでしたので，すぐ8251Aのほうへ手が伸びてしまいました．

現在では Z80 SIO も非常に安くなり，手軽に使えるようになりました．このSIOは使えば使うほど，すばらしさがわかってき，よくこれほど複雑なものを考え出すことができたものだ，と感心せざるを得ません．

さて前置きはこれまでにして，ここではZ80 SIOが持っている機能を列挙し，必要に応じて説明を加えることにします．なお，これ以降，とくに必要な場合を除いて，Z80 SIOのことを単にSIOと呼ぶことにします．

■互いに独立な全2重チャネルを2個持っている

SIOはチャネルAおよびチャネルBという名の全2重チャネルを持っており，それぞれは独立に動作することができます．各チャネルは全2重で動作しますので，トランスミッタとレシーバも独立かつ同時に動けます．

■データ転送速度

- Z80 SIO（2.5MHz）――0～550kBPS
- Z80A SIO（4MHz）――0～880kBPS

SIOはスペックにより，SIOに入力されるシステム・クロックの周波数は，シリアル・データのボーレートの4.5倍以上と決まっています．したがって，次の割り算を行うことにより，SIOの最高ボーレートを求めることができます．

2.5MHz÷4.5＝555.55〔kBPS(Z80 SIO)〕
4.0MHz÷4.5＝888.88〔kBPS(Z80A SIO)〕

■受信バッファは4重化されており，送信バッファは2重化されている

受信バッファおよび送信バッファは双方ともFIFO（First In First Out）形式です．受信バッファが長いということは，CPUの都合により，SIOから受信データを読み取るのが遅れたとしても，レシーバ・オーバランになる機会を最小にとどめるために有効です．

■非同期通信における機能

- キャラクタ長――5，6，7，8ビット
- ストップ・ビット――1，1½，2ビット
- パリティ・ビット――奇偶パリティ付加が可
- 送受信クロックの倍率――×1，×16，×32，×64
- ブレーク発生および検出機能
- パリティ・エラー，オーバラン・エラー，フレーミング・エラーの検出機能

■同期通信方式の機能

- 内部同期，および外部同期が可能
- シンク・キャラクタ・レジスタを2個持っている
- シンク・キャラクタの挿入を自動的にできる
- CRC発生およびチェック機能を持っている

内部同期，外部同期ということばについてですが，ブロック・データの受信時に，シンク・キャラクタの検出をSIO内部でやることを内部同期，SIOの外でやることを外部同期と呼びます．

■SDLC方式の機能

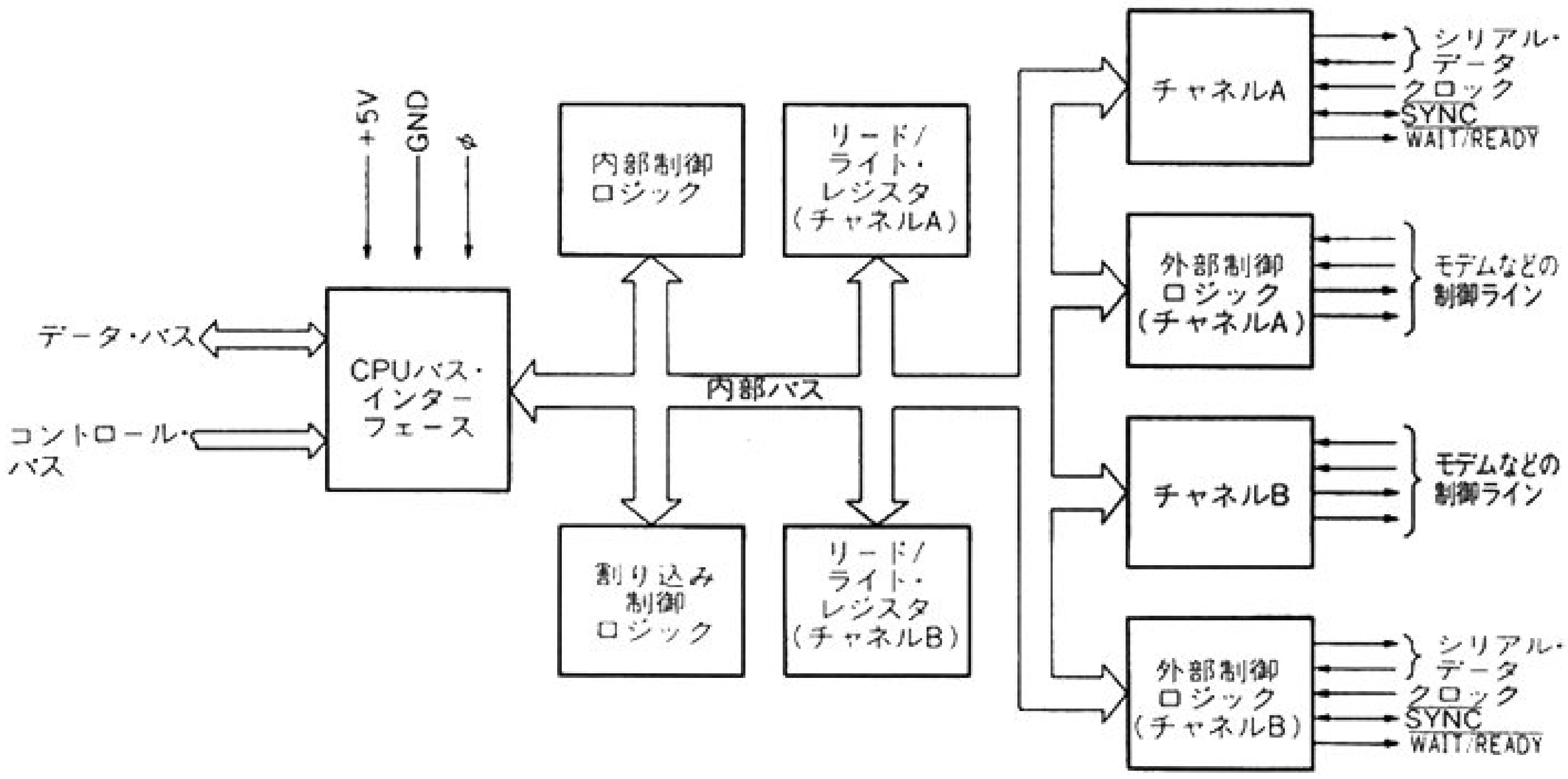
- アボート・シーケンスの発生および検出ができる
- ゼロ・インサーション，およびゼロ・デリーション機能を持っている
- メッセージ・フレーム間に，自動的にフラグを挿入できる
- アドレス・フィールドの認識機能
- Iフィールドの最後の端数ビット（Residue）の処理ができる
- CRC発生およびチェック機能を持っている

■両チャネルに対してモデムなどの外部装置を制御するための制御線が設けられている

SIOには4本の外部装置制御用のライン，$\overline{\mathrm{RTS}}$(A/B)，$\overline{\mathrm{DTR}}$(A/B)，$\overline{\mathrm{CTS}}$(A/B)，$\overline{\mathrm{DCD}}$(A/B)があり，前の二つは出力であり，後の二つは入力です．$\overline{\mathrm{CTS}}$(A/B)および$\overline{\mathrm{DCD}}$(A/B)の状態変化を割り込みにより監視することができます．

■CRCチェック機能として，CRC-16（同期通信用），およびCRC-CCITT（SDLC用）の双方が設けられている

〈図7〉
Z80 SIOの構成



■ディジー・チェイン方式の優先ベクトル割り込み制御ロジックを内蔵している

Z80 SIOの構成

図7にSIOの構成を示します．この図からわかるように，SIOはCPUバス・インターフェース，内部制御ロジック，割り込み制御ロジック，リード／ライト・レジスタ(チャネルA，チャネルB)，それにA，B両チャネルならびに各チャネルに対する外部制御ロジックからなります．

CPUバス・インターフェースは，CPUとSIO間のコントロール情報（コマンドやステータス）やデータのやりとりを制御します．内部制御ロジックはその名のとおり，SIO内部の制御を行いますが，SIO内部のロジックを見たことがありませんので，これ以上のことはわかりません．

割り込み制御ロジックは，ご存知のZ80ファミリのディジー・チェイン式，優先ベクトル割り込み方式をサポートします．各チャネルごとにあるリード／ライト・レジスタは，各チャネルの動作を制御するコントロール・レジスタです．

ライト・レジスタはCPUから書き込まれるコマンドを保持し，リード・レジスタにはSIOのステータスがセットされます．ライト・レジスタは8個（WR0～WR7)，リード・レジスタは3個(RR0～RR2)あります．

チャネルAおよびチャネルBは，互いに独立に動作できる全2重シリアル・インターフェースです．外部制御ロジックも，チャネルA，チャネルBに対して専用に設けられており，SIOとインターフェースする外部装置，たとえばモデムなどの制御，およびステータスの監視を行います．

図8に各チャネルの構成を示します．この構成は一般的なシリアル・インターフェースの構成と同じですが，SIOはCRCジェネレータおよびチェッカを内蔵しており，かつSDLC制御のために，ゼロ・インサーションを行うゼロ・インサータおよびゼロ・デリーションを行う，ビット・ストリッパが設けられています．

レシーバの動作

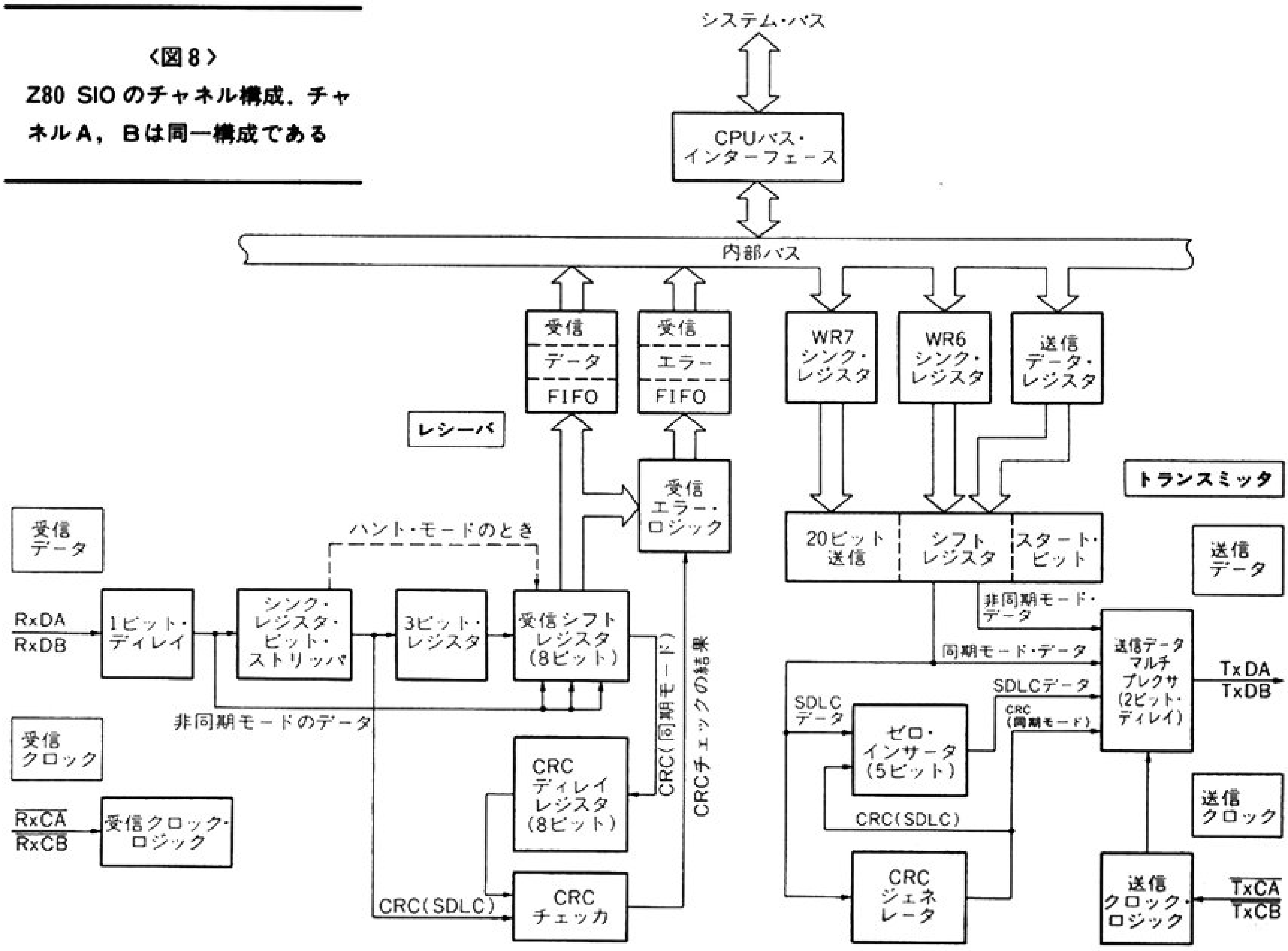
レシーバはFIFO形式の三つのバッファ・レジスタと8ビット・シフトレジスタを持っています．このためCPUは，高速データ転送のブロック開始時の割り込みサービスに時間の余裕を得ることができます．また，レシーバはデータに対してだけではなく，パリティ・エラーやフレーミング・エラーなどの個々のデータに付随したステータスをキューイングするために，もう1本のFIFO(受信エラーFIFO)を持っています．

入力データ(RxDA/B)は動作モードおよびキャラクタ長に応じて，いくつかのルートの中の一つを通ります．非同期式モードの場合には，キャラクタ長が7または8ビットであるときには，入力データは3ビット・バッファに入り，キャラクタ長が5または6ビットの場合には，3ビット・バッファをバイパスし，8ビットの受信シフトレジスタに入ります．

同期式モードの場合には，その時点の受信状態により，入力データの経路は異なります．同期式モードの受信動作はハント(Hunt)・フェーズ（段階）から始まります．このフェーズでは，レシーバは入力データ・ストリーム（データの流れ）をサーチし，ライト・レジスタにプログラムされたシンク・キャラクタ(SDLCの場合にはオープニング・フラグ)と一致するのを待ちます．

SIOがシングル・シンク・モードにプログラミングされているときには，WR7にストアされたシンク・キ

〈図8〉
Z80 SIO のチャネル構成．チャネルA，Bは同一構成である



ャラクタと一致するのを待ち，ダブル・シンク・モードにプログラミングされているときには，あい続く二つのキャラクタが WR6 および WR7 にストアされている，シンク・キャラクタと一致するのを待ちます．

シンク・キャラクタとの一致が検出されると，データ・ラインの同期が確立され，以降の入力データはシンク・レジスタをバイパスし，3ビット・バッファに入ります．

SDLC モードの場合には，入力データは受信シンク・レジスタを通過し，このレジスタは常に受信データ・ストリームを監視しており，ゼロ・デリーションの必要があるか否かを見ています．

受信データ・ストリーム中に連続した5個の"1"を検出すると，このレジスタは注意力を次のビットに集中します．そして次のビットが"0"である場合にはこれを削除し(ゼロ・デリーション)，"1"である場合にはさらに次のビットに注意を払います．

次のビットが"0"である場合には，フラグ("01111110")が受信されたものと解釈し，"1"である場合にはアボート・シーケンス(7個以上の連続した"1")が受信されたものと解釈します．

必要に応じてゼロ・デリーションされたデータは3ビット・バッファに入り，その後受信シフトレジスタに転送されます．同期式モードの場合と同様に SDLC モードの場合も，ハント・フェーズから受信動作が開始され，受信シフトレジスタの内容が WR7 の内容と一致することがチェックされますが，いったんフラグが認識されるとデータはキャラクタ長とは関係なく同じ経路をたどります．

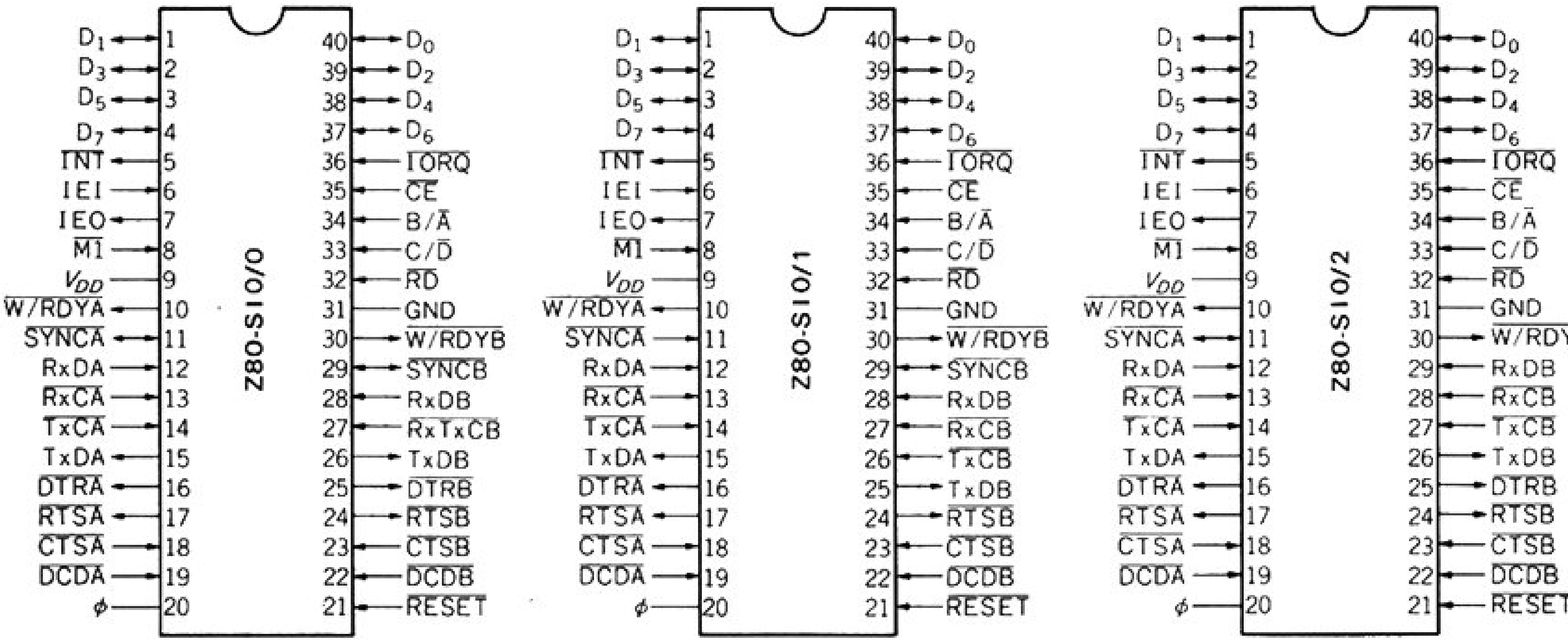
SDLC モードおよび同期式モードでは，同一の CRC チェッカが用いられますが，各モードにおけるデータ経路は異なります．Bisync プロトコルでは，ある特定のキャラクタを CRC 計算に含めるか否かを CPU が判断する必要があります．

この判断に要する時間をかせぐために，SIO は同期式モードのデータに対して8ビット・タイムのディレイを設けています．SDLC モードでは SIO 自体にこの判断の機能があるため，ディレイの必要はありません．

トランスミッタの動作

トランスミッタは8ビット送信データ・レジスタ，および20ビット・シフトレジスタを持っています．送信データ・レジスタは内部データ・バスからデータが

〈図9〉 Z80 SIO/0~2のピン配列（ピン㉕~㉙が異なる）



ロードされ，20ビット・シフトレジスタには WR6，WR7および送信データ・レジスタからデータがロードされます．

WR6は同期式モードではシンク・キャラクタを保持し，SDLCモードではステーション・アドレスを保持します．WR7はダブル・シンク・モードにおける，第2のシンク・キャラクタまたは SDLC におけるフラグを保持します．

同期式モードでは，WR6およびWR7の内容はデータ・ブロックの開始時，およびデータ・ブロックの途中でトランスミッタ・アンダーラン状態（送信すべき時間に，データの用意ができなかった場合）が生じたときに，アイドル・シンク・キャラクタとして送信シフトレジスタにロードされます．SDLCモードでは，フレームの送信開始時に，フラグ（WR7）が送信シフトレジスタにロードされます．

非同期式モードの場合には，送信シフトレジスタの内容にスタート・ビットおよびストップ・ビットが付加され，送信データ・マルチプレクサに向けてシフト・アウトされます．同期式モードのデータは送信データ・マルチプレクサおよび CRC ジェネレータに向けてシフト・アウトされます．

SDLCモードのデータはゼロ・インサータを通じてシフト・アウトされます．このゼロ・インサータはフラグ，およびアボート・シーケンスが送信されるときには，ディスエーブルされます．CRCジェネレータの出力もやはりゼロ・インサータを経由します．

Z80 SIOのピン機能

SIOは40ピン DIP に実装されています．図9にSIOのピン配置を示します．SIOはピン・リミットの関係から3種類のボンディング・オプションを持っています．

ボンディング・オプションは文字"SIO"に続くスラッシュ（/）と番号（0~2）により示されます．ボンディング・オプションにより，ピン配置が異なるのは，ピン㉕~㉙であり，チャネルB関係の信号です．

チャネルAとチャネルBはまったく同じ機能を持ちますので，本来ならばチャネルBに関しても，$\overline{\mathrm{RxCB}}$，$\overline{\mathrm{TxCB}}$，$\overline{\mathrm{SYNCB}}$，ならびに $\overline{\mathrm{DTRB}}$ 信号をピン上に出さなければならないのですが，ピン数が足りないため，これら4本のうちの1本を切り捨てる必要があったのです．

図10に各ボンディング・オプション間の関係を示します．SIO/0では $\overline{\mathrm{TxCB}}$ と $\overline{\mathrm{RxCB}}$ を1本のピンにボンディングして，$\overline{\mathrm{RxTxCB}}$ としています．つまりチャネルBの受信クロックと送信クロックを共通信号としたものです．

SIO/1では $\overline{\mathrm{DTRB}}$ がピン上に出ておらず，SIO/2では $\overline{\mathrm{SYNCB}}$ が姿を消しています．したがってユーザは設計時に自分のシステムとうまく適合する，ボンディング・オプションのSIOを選択する必要があります．

次にSIOの各ピンの機能を説明します．図11に各ピンの機能をまとめます．

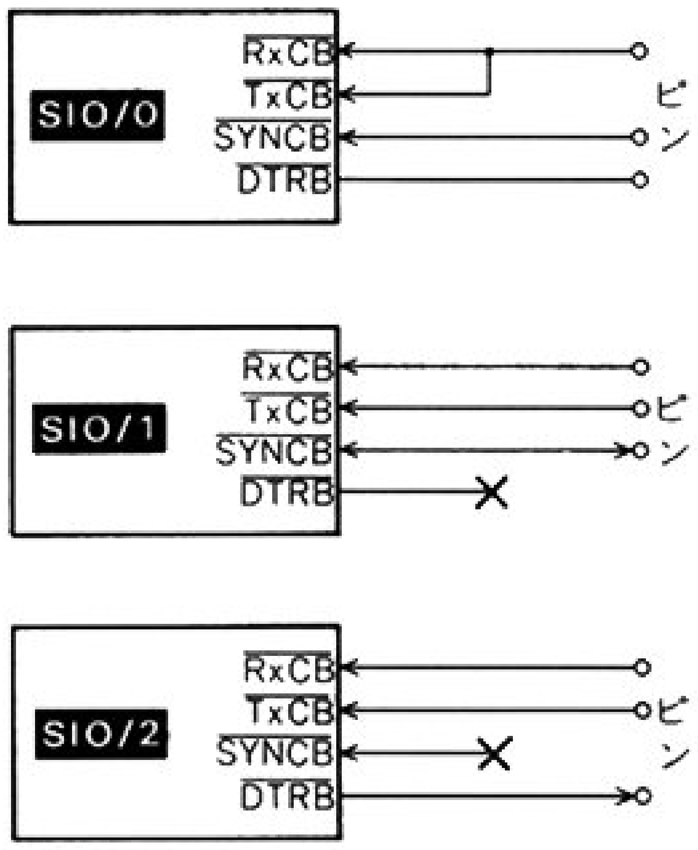
■D_0~D_7：システム・データ・バス（双方向性）

SIOはこのシステム・データ・バスを通じて，CPUとデータ，コマンドならびにステータスのやりとりをします．

■B/$\overline{\mathrm{A}}$：チャネル・セレクト（入力）

この信号により，SIOのチャネルを選択します．この信号が"L"であるとチャネルAが選択され，"H"であるとチャネルBが選択されます．通常は CPU の

〈図10〉 Z80 SIOのボンディング・オプション



〈図12〉 SIO 内部レジスタのアドレッシング．$B/\overline{A}=A_1$, $C/\overline{D}=A_0$ とした場合

$B/\overline{A}=A_1$	$C/\overline{D}=A_0$	アドレス (16進)	セレクトされるレジスタ	コメント
L	L	×0	チャネルAデータ	送受信データ・レジスタ
L	H	×1	チャネルAコントロール	ライト・レジスタ リード・レジスタ
H	L	×2	チャネルBデータ	送受信データ・レジスタ
H	H	×3	チャネルBコントロール	ライト・レジスタ リード・レジスタ

アドレス・バスのビット0 (A_0) またはビット1 (A_1) が接続されます．

■$C/\overline{D}$：コマンド／データ（入力）

この信号はSIOに対するリード／ライト操作の対象がコマンド・レジスタ（ライト・レジスタ，またはリード・レジスタ）であるのか，データ・レジスタ（送信データ・レジスタ，または受信データ・レジスタ）であるのかを指定します．

この信号が“L”であるとデータ・レジスタとなり，“H”であるとコマンド・レジスタとなります．そして通常はこの信号には CPU のアドレス・バスのビット0 (A_0) またはビット1 (A_1) を接続します．

アクセスする SIO 内部のレジスタは，この信号と前述の $C/\overline{D}$ により決定されます．$B/\overline{A}=A_1$, $C/\overline{D}=A_0$ とした場合のZ80内部のレジスタのアドレッシングを図12に示します．

■$\overline{CE}$：チップ・イネーブル（入力，“L”で有効）

この信号を“L”にすると当該SIOがセレクトされます．この信号には当該 SIO に対するアドレス・デコーダの出力が接続され，SIOの内部レジスタに対してリード／ライト操作を行うときには，“L”にする必要があります．

■ϕ：システム・クロック（入力）

SIO に対するシステム・クロック入力であり，Z80 CPUのシステム・クロックをそのまま用いることができます．このクロックの周波数の最大値は，Z80 SIO では2.5MHz, Z80A SIO では4MHzです．またこのクロックの周波数はシリアル・データのボーレートの4.5倍以上である必要があります．

■$\overline{M1}$：マシン・サイクル1（入力，“L”で有効）

この信号には Z80 CPU の $\overline{M1}$ 信号をそのまま接続します．この信号と $\overline{IORQ}$ が同時に“L”になると，CPU が割り込み認識シーケンスを実行していることを示し，当該 SIO より高い割り込み優先順位を持つペリフェラルが割り込み要求を出していないときには，当該 SIO はデータ・バス上に割り込みベクトルを乗せます．

■$\overline{IORQ}$：I/O リクエスト（入力，“L”で有効）

この信号には Z80 CPU の $\overline{IORQ}$ 信号をそのまま接続します．SIO の内部レジスタにアクセスするときには，前述の $\overline{CE}$ およびこの信号がともに“L”でなければなりません．この信号は Z80 CPU の割り込み認識シーケンスにおいても用いられます．

■$\overline{RD}$：リード（入力，“L”で有効）

この信号も Z80 CPU の $\overline{RD}$ 信号に直接接続されます．この信号が“L”であり，かつ $\overline{CE}$ および $\overline{IORQ}$ がともに“L”であると，当該 SIO に対してリード命令が出されていることを示します．対象とするチャネルは $B/\overline{A}$ により指定され，$C/\overline{D}$ によりコマンドであるのかデータであるのかの選択が行われます．SIO には CPU の $\overline{WR}$ 信号は入力されませんので，この $\overline{RD}$ が“H”であるときには SIO に対してライト命令が出されていることを示します．

■$\overline{RESET}$：リセット（入力，“L”で有効）

この信号が“L”であるとA，B両チャネルのレシーバおよびトランスミッタがディスエーブルされ，シリアル・データ・ライン (TxDA, TxDB) はマーク状態になります．このリセット信号が出された後のコントロール・レジスタ（ライト・レジスタ）の値は保証されていませんので，必要なすべてのライト・レジスタに対して，コマンドを書き直す必要があります．このリセット信号は最低1システム・クロック (ϕ) 期間，“L”にする必要があります．

〈図11〉 Z80 SIOのピン機能のまとめ

ピン名	信号名	I/O	有効レベル	説明
D_0～D_7	システム・データ・バス	双方向性		SIOはこのバスを通じて，CPUとデータ，コマンドならびにステータスのやりとりを行う
B/$\overline{A}$	チャネル・セレクト	入力		"L"：チャネルA，"H"：チャネルB
C/$\overline{D}$	コマンド/データ	入力		"L"：データ，"H"：コマンド B/$\overline{A}$ ｜ C/$\overline{D}$ ｜ セレクトされるレジスタ L ｜ L ｜ チャネルAデータ L ｜ H ｜ チャネルAコントロール H ｜ L ｜ チャネルBデータ H ｜ H ｜ チャネルBコントロール
$\overline{CE}$	チップ・イネーブル	入力	L	この信号が"L"であると，当該SIOがイネーブルされる
ϕ	システム・クロック	入力		Z80 CPUのシステム・クロックをそのまま接続する (Z80 SIO：最高2.5MHz，Z80A SIO：最高4MHz)
$\overline{M1}$	マシン・サイクル1	入力	L	Z80CPUは割り込み認識シーケンス時に，この$\overline{M1}$と$\overline{IORQ}$をともに"L"にする．Z80 CPUの$\overline{M1}$をそのまま接続する
$\overline{IORQ}$	I/Oリクエスト	入力	L	Z80 CPUの$\overline{IORQ}$をそのまま接続する．SIOに対するリード/ライト操作はこの$\overline{IORQ}$と$\overline{CE}$がともに"L"であるときに行われる．CPUの割り込み認識シーケンスにもこの信号が"L"になる
$\overline{RD}$	リード	入力	L	Z80 CPUの$\overline{RD}$をそのまま接続する．$\overline{CE}$および$\overline{IORQ}$がともに"L"であるときに，この信号が"L"であるとSIOに対するリード動作が行われていることを示し，"H"であるとライト動作が行われていることを示す
$\overline{RESET}$	リセット	入力	L	SIOに対するリセット信号．この信号のパルス幅は最低1システム・クロック期間必要である
IEI	インタラプト・イネーブル・イン	入力	H	この信号が"H"であると，当該SIOより高い優先順位を持つペリフェラルが，割り込み要求を出していないことを示す
IEO	インタラプト・イネーブル・アウト	出力	H	この信号は，当該SIOより高い優先順位を持つペリフェラルが割り込み要求を出していない場合，および当該SIOが割り込み要求を出していないときに"H"となる
$\overline{INT}$	割り込み要求	出力	L	当該SIOがCPUに対して割り込み要求を出すときに，この信号を"L"にする．この信号の出力はオープン・ドレイン型であるのでワイヤードOR接続が可能
$\overline{W/RDYA}$ $\overline{W/RDYB}$	ウェイト/レディ	出力	ウェイト:L レディ時はプログラム可	ウェイトとプログラムされたときには，SIOが送信データを受け取る状態にないとき，または受信データの用意ができていないときに，この信号は"L"となる．この場合，出力はオープン・ドレイン型となるのでワイヤードOR接続が可能である． レディとプログラムされたときには，Z80 DMAに対するレディ信号となる．
$\overline{CTSA}$ $\overline{CTSB}$	クリヤ・ツー・センド	入力	L	モデムの出力，CTSを接続する．この信号が"L"であるとモデムが送信データを受け付けられる状態にあることを示す
$\overline{DCDA}$ $\overline{DCDB}$	データ・キャリヤ・デテクト	入力	L	モデムの出力，DCDを接続する．この信号が"L"であるとモデムが受信キャリヤを検出していることを示す
RxDA RxDB	受信データ	入力	"H"：1 "L"：0	SIOに対するシリアル・データ入力
TxDA TxDB	送信データ	出力	"H"：1 "L"：0	SIOのシリアル・データ出力
$\overline{RxCA}$ $\overline{RxCB}$	受信クロック	入力		SIOに対する受信クロック入力であり，Z80 CTCの出力またはモデムの受信クロックを接続する
$\overline{TxCA}$ $\overline{TxCB}$	送信クロック	入力		SIOに対する送信クロック入力であり，Z80 CTCの出力またはモデムの送信クロックを接続する
$\overline{RTSA}$ $\overline{RTSB}$	リクエスト・ツー・センド	出力	L	SIOからモデムに対して，データを送信したいときに，この信号を"L"にする
$\overline{DTRA}$ $\overline{DTRB}$	データ・ターミナル・レディ	出力	L	モデムに対して，データ・ターミナルがレディ状態にあることを示す
$\overline{SYNCA}$ $\overline{SYNCB}$	同期確立	入出力	L	同期モードにおいて，外部同期モードとプログラミングされているときには，この信号は入力となる．外部回路により同期の確立を検出したときに，この信号を"L"にする． 内部同期の場合には出力となり，SIO内部で同期確立を検出したときに，この信号は"L"となる

■IEI：インタラプト・イネーブル・イン（入力，"H"で有効）

これは，Z80ファミリのディィジー・チェイン式の優先割り込み制御を行うための信号であり，この信号が"H"であると，当該SIOより割り込み優先順位の高いペリフェラルが割り込み要求を出していないことを示します．

当該SIOがシステム中で最高の割り込み優先順位を持っている場合には，このIEI入力を＋5Vにプルアップします．

■IEO：インタラプト・イネーブル・アウト（出力，"H"で有効）

この信号はIEIとともに，Z80ファミリのディジー・チェイン式優先割り込み制御を行うためのものであり，当該SIOより低い割り込み優先度を持つペリフェラルのIEIに接続されます．

SIOは自分よりも高い割り込み優先順位を持つ他のペリフェラルが割り込み要求を行っているとき(IEI＝"L")，および自分自身が割り込み要求を行っているときには，このIEO信号を"L"にします．

■$\overline{\text{INT}}$：割り込み要求（出力，"L"で有効）

SIOがCPUに対して割り込み要求を行うときに，この信号を"L"にします．この信号の出力型式は，オープン・ドレイン型になっていますので，他のペリフェラルの$\overline{\text{INT}}$信号とワイヤードOR接続ができるようになっています．

■$\overline{\text{W/RDYA}}$, $\overline{\text{W/RDYB}}$：ウェイト／レディ（出力）

この出力信号はSIOのプログラミングにより，ウェイト機能として用いたり，レディ機能として用いたりすることができます．ウェイト機能とプログラミングしたときには，出力はオープン・ドレイン型となり，他のウェイト信号とワイヤードOR接続してCPUの$\overline{\text{WAIT}}$入力に接続します．

この場合にはSIOのデータ・レジスタに対してデータをライトしたとき，送信データ・バッファがバッファ・エンプティでないときには，この出力が"L"となりCPUに対してウェイトをかけます．

また，SIOのデータ・レジスタに対してリード命令を出したときに，受信データの用意ができていないときにもこの信号は"L"になりCPUを待たせます．

レディ機能としてプログラミングしたときには，この信号はZ80 DMA用のレディ信号の役目を果たします．この場合には，この信号の出力はTTLコンパチブルとなります．

レディ機能として用いるときには，この信号はトランスミッタ・バッファ・エンプティまたはレシーバにデータが用意されたときに有効レベルとなり，1バイトのデータが転送されると(DMAデータ転送)，無効レベルに戻ります．

■$\overline{\text{CTSA}}$, $\overline{\text{CTSB}}$：クリヤ・ツー・センド（入力，"L"で有効）

この信号は通常RS-232-Cインターフェースの$\overline{\text{CTS}}$入力として用いられます．SIOではこの信号の状態が変化すると（H→L，L→H)，外部ステータス割り込みという割り込みを発生することができます．

また，SIOを"オート・イネーブル"とプログラミングしたときには，この$\overline{\text{CTS}}$入力が"L"にならないとトランスミッタはイネーブルされません（たとえイネーブル・トランスミッタ・コマンドを出したとしても）．

■$\overline{\text{DCDA}}$, $\overline{\text{DCDB}}$：データ・キャリヤ・デテクト（入力，"L"で有効）

この信号は通常RS-232-Cインターフェースの$\overline{\text{DCD}}$入力として用いられます．$\overline{\text{CTSA}}/\overline{\text{CTSB}}$の場合と同様に，この信号の状態が変化したときに，外部ステータス割り込みを発生することができます．またSIOをオート・イネーブルとプログラミングしたときには，この信号が"L"にならないとレシーバはイネーブルされません（たとえイネーブル・レシーバ・コマンドを出したとしても）．

■RxDA, RxDB：受信データ（入力）

SIOに対するシリアル・データの入力であり，データ・ビットが"0"のときは"L"，"1"のときは"H"となります．

■TxDA, TxDB：送信データ（出力）

SIOのシリアル・データ出力であり，データ・ビットが"0"のときは"L"となり，"1"のときは"H"となります．SIOはリセットされるとこの出力の状態を"H"(マーク）状態にします．

■$\overline{\text{RxCA}}$, $\overline{\text{RxCB}}$：受信クロック（入力）

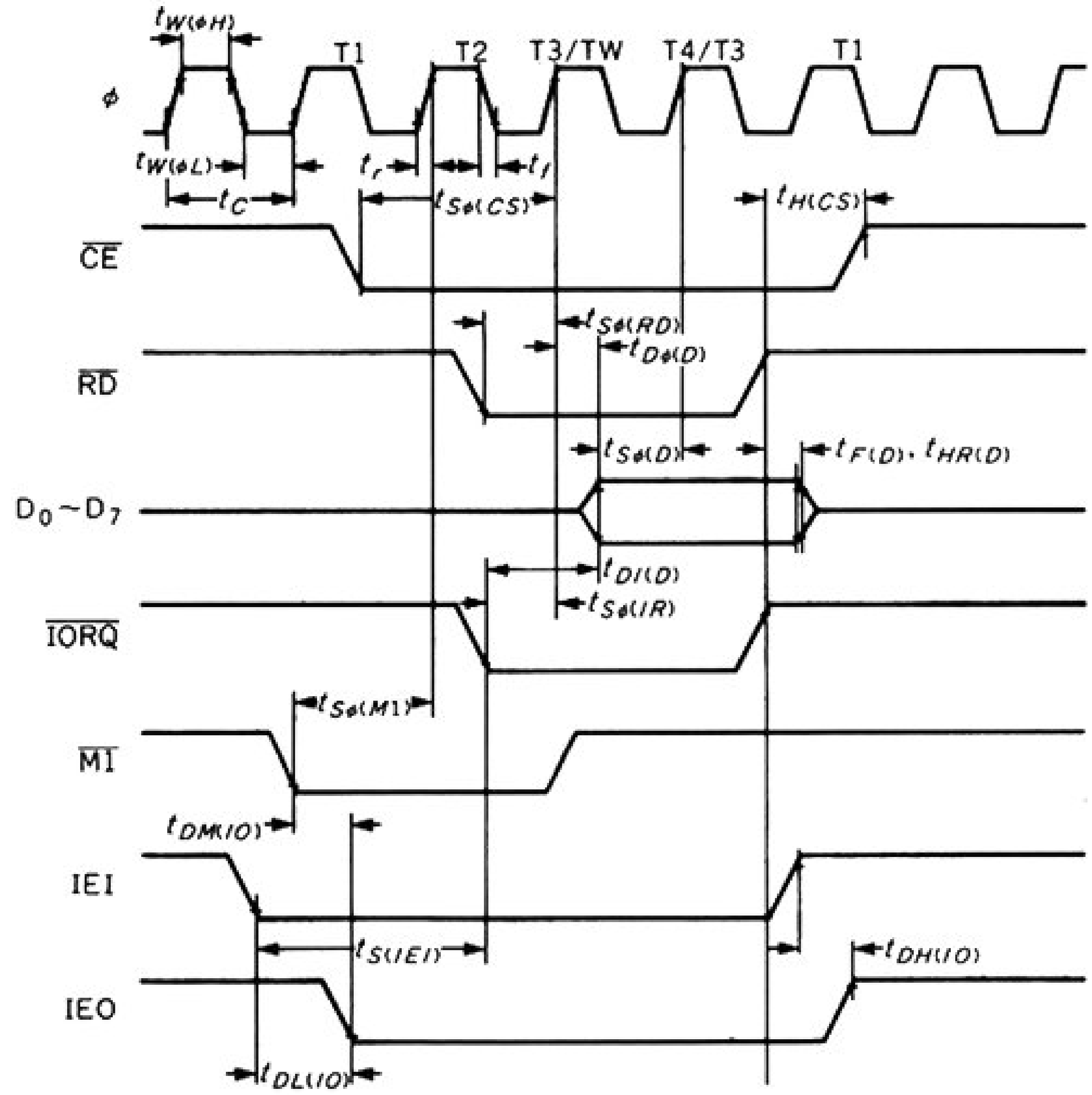
SIOに対する受信クロック入力であり，このクロックの立ち上がり時点に受信データ・ラインの状態がサンプリングされます．この受信クロックに対しては，"L"期間および"H"期間の最小時間が規定されているだけであり，その値は180nsですので，Z80 CTCの出力をそのまま使用することができます．

ただし同期モード，またはSDLCモードにおいてモデムとインターフェースするときには，モデムから

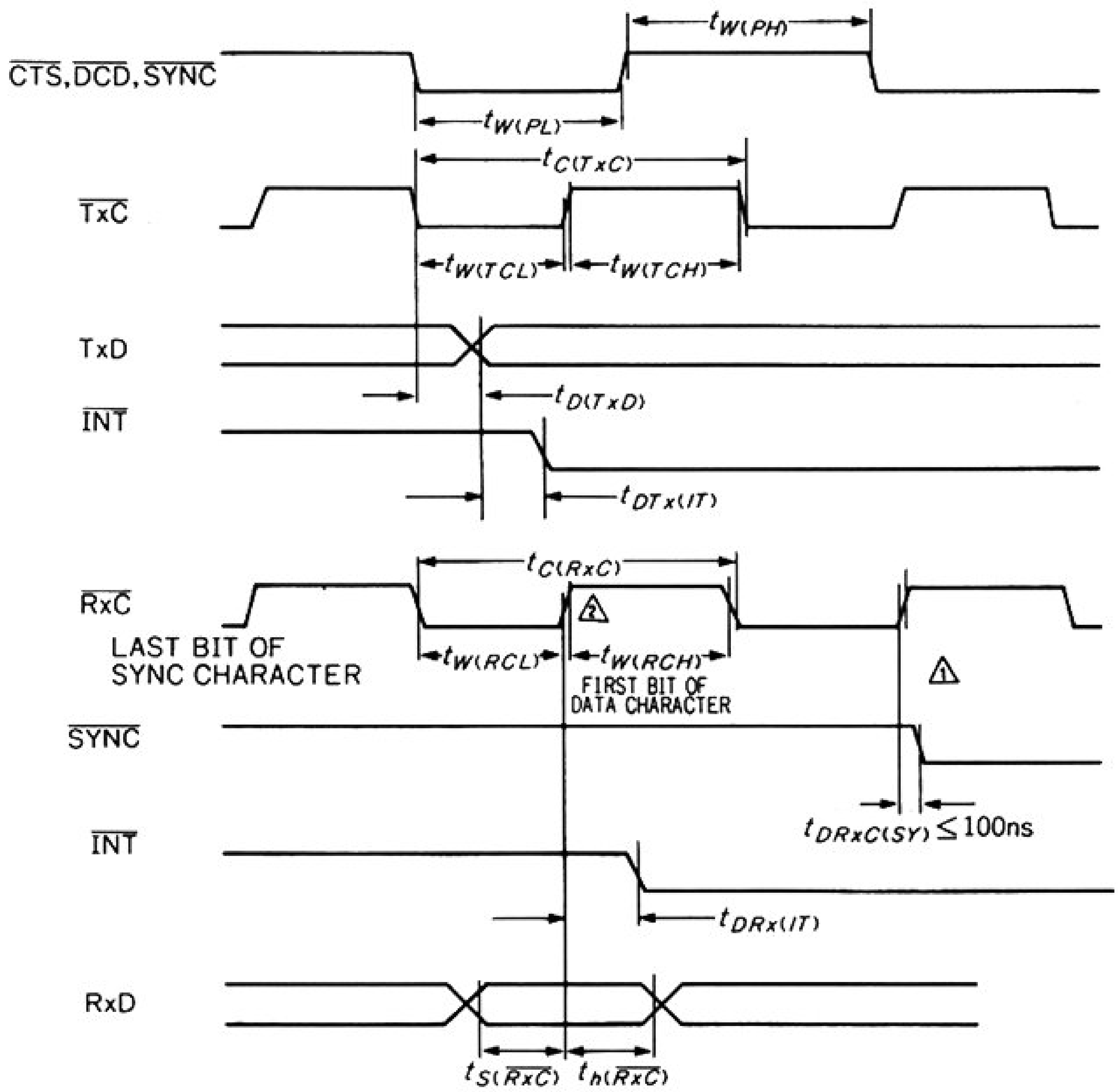
<図13>
Z80 SIOの特性
(ザイログ社のデータ・シートより)

●AC 特性●

(T_a = 0°C～70°C, V_{CC} = +5 V ± 5 %)



Signal	Symbol	Parameter	Z80-SIO Min	Z80-SIO Max	Z80A-SIO Min	Z80A-SIO Max	Unit
φ	$t_C(\phi)$	Clock Period	400	4000	250	4000	ns
	$t_W(\phi H)$	Clock Pulse Width, clock HIGH	170	2000	105	2000	ns
	$t_W(\phi L)$	Clock Pulse Width, clock LOW	170	2000	105	2000	ns
	t_r, t_f	Clock Rise and Fall Times	0	30	0	30	ns
	t_H	Any Unspecified Hold Time for setup times specified below	0		0		ns
$\overline{CE}$, $\overline{B/A}$ C/D, $\overline{IORQ}$	$t_{S\phi}(CS)$	Control Signal Setup Time to rising edge of φ during Read or Write Cycle	160		145		ns
D_0-D_7	$t_{D\phi}(D)$	Data Output Delay from rising edge of φ during Read Cycle		240		220	ns
	$t_{S\phi}(D)$	Data Setup Time to rising edge of φ during Write or M1 Cycle	50		50		ns
	$t_{DI}(D)$	Data Output Delay from falling edge of $\overline{IORQ}$ during INTA Cycle		340		160	ns
	$t_F(D)$	Delay to Floating Bus (output buffer disable time)		230		110	ns
IEI	$t_S(IEI)$	IEI Setup Time to falling edge of $\overline{IORQ}$ during INTA Cycle	200		140		ns
IEO	$t_{DH}(IO)$	IEO Delay Time from rising edge of IEI (after 'ED' decode)		150		100	ns
	$t_{DL}(IO)$	IEO Delay Time from falling edge of IEI		150		100	ns
	$t_{DM}(IO)$	IEO Delay Time from falling edge of $\overline{M1}$ (interrupt occurring just prior to M1)		300		190	ns
M1	$t_{S\phi}(M1)$	$\overline{M1}$ Setup Time to rising edge of φ during INTA or M1 Cycle	210		90		ns
$\overline{RD}$	$t_{S\phi}(RD)$	$\overline{RD}$ Setup Time to rising edge of φ during Read or M1 Cycle	240		115		ns



注1；$\overline{\text{SYNC}}$入力はシンク・キャラクタの最後の $\overline{\text{RxC}}$ の立ち上がりから2番目の $\overline{\text{RxC}}$ の立ち上がり時に"L"にする必要がある

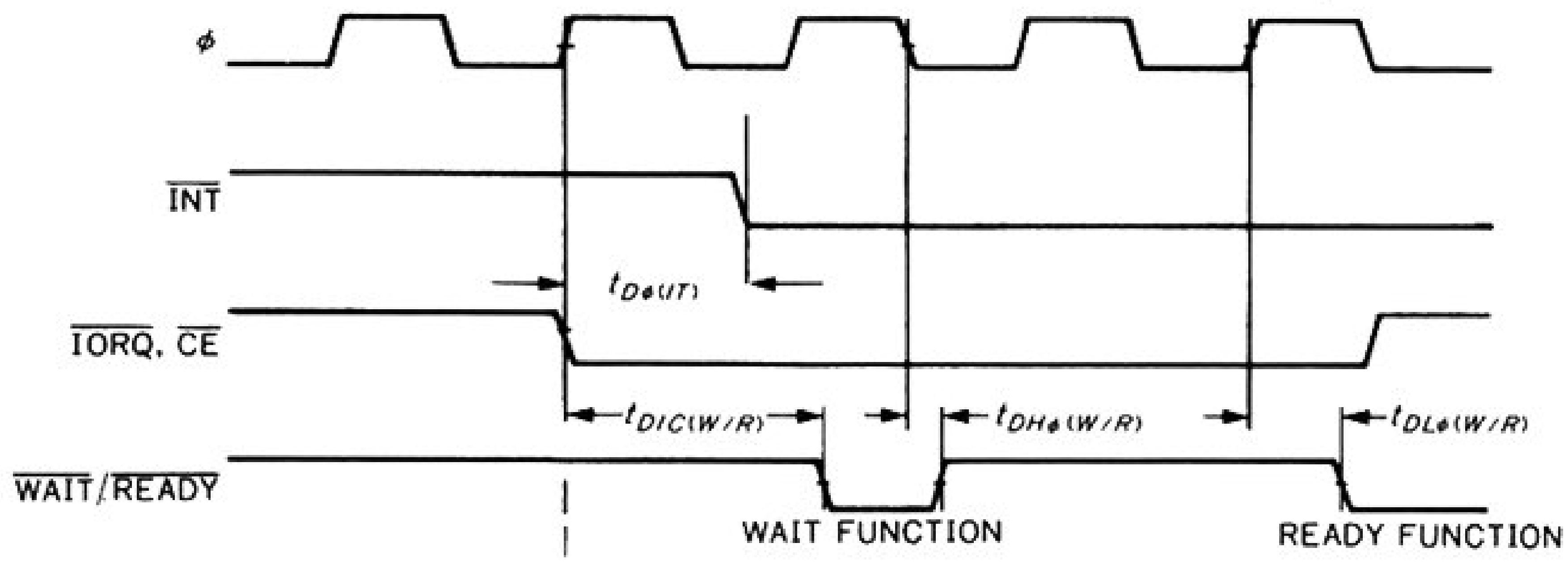
注2；データ・キャラクタの組み立て(シリアル→パラレル)は，シンク・キャラクタの最後のビットの次の受信クロックから開始される

Signal	Symbol	Parameter	Z80-SIO Min	Z80-SIO Max	Z80A-SIO Min	Z80A-SIO Max	Unit
$\overline{\text{INT}}$	$t_{DRx}(IT)$	$\overline{\text{INT}}$ Delay Time from rising edge of RxC	10	13	10	13	φ periods
	$t_{DTx}(IT)$	$\overline{\text{INT}}$ Delay Time from transition of Xmit Data Bit	5	9	5	9	φ periods
$\overline{\text{CTSA}}$, $\overline{\text{CTSB}}$, $\overline{\text{DCDA}}$, $\overline{\text{DCDB}}$, $\overline{\text{SYNCA}}$, $\overline{\text{SYNCB}}$	$t_w(PH)$	Minimum HIGH Pulse Width for latching state into register and generating interrupt	200		200		ns
	$t_w(PL)$	Minimum LOW Pulse Width for latching state into register and generating interrupt	200		200		ns
$\overline{\text{SYNCA}}$, $\overline{\text{SYNCB}}$	$t_{DL}(SY)$	Sync Pulse Delay Time from rising edge of $\overline{\text{RxC}}$, Output Modes	4	7	4	7	φ periods
	$t_{DRxC}(SY)$	Sync Pulse Delay Time from rising edge of $\overline{\text{RxC}}$ External Sync Mode		100		100	ns
$\overline{\text{TxCA}}$, $\overline{\text{TxCB}}$	$t_c(TxC)$	Transmit Clock Period	400	∞	400	∞	ns
	$t_w(TCH)$	Transmit Clock Pulse Width, clock HIGH	180	∞	180	∞	ns
	$t_w(TCL)$	Transmit Clock Pulse Width, clock LOW	180	∞	180	∞	ns
TxDA, TxDB†	$t_D(TxD)$	TxD Output Delay from falling Edge of $\overline{\text{TxC}}$ (x1 Clock Mode)		400		300	ns
$\overline{\text{RxCA}}$, $\overline{\text{RxCB}}$	$t_c(RxC)$	Receive Clock Period	400	∞	400	∞	ns
	$t_w(RCH)$	Receive Clock Pulse Width, clock HIGH	180	∞	180	∞	ns
	$t_w(RCL)$	Receive Clock Pulse Width, clock LOW	180	∞	180	∞	ns
RxDA, RxDB†	$t_s(RxC)$	Setup Time to rising edge of $\overline{\text{RxC}}$, x1 mode	0		0		ns
	$t_h(RxC)$	Hold Time from rising edge of $\overline{\text{RxC}}$, x1 mode	140		140		ns

注3；どのモードにおいても，システム・クロックの周波数は，ボーレートの4.5倍以上でなければならない

注4；$\overline{\text{RESET}}$信号は，最低1システム・クロック・サイクル必要である

●AC 特性(つづき)●



Signal	Symbol	Parameter	Z80-SIO Min	Z80-SIO Max	Z80A-SIO Min	Z80A-SIO Max	Unit
INT	$t_D\phi(IT)$	INT Delay Time from rising edge of ϕ		200		200	ns
WAIT/READY	$t_DIC(W/R)$	WAIT/READY Delay Time from IORQ or CE in Wait Mode		180		130	ns
	$t_DH\phi(W/R)$	WAIT/READY Delay Time from falling edge of ϕ, WAIT/READY HIGH, Wait Mode		150		130	ns
	$t_DRx(W/R)$	WAIT/READY Delay Time from rising edge of RxC Data Bit, Ready Mode	10	13	10	13	ϕ periods
	$t_DTx(W/R)$	WAIT/READY Delay Time from center of Transmit Data Bit, Ready Mode	5	9	5	9	ϕ periods
	$t_DL\phi(W/R)$	WAIT/READY Delay Time from rising edge of ϕ, WAIT/READY LOW, Ready Mode		120		120	ns

●DC 特性●

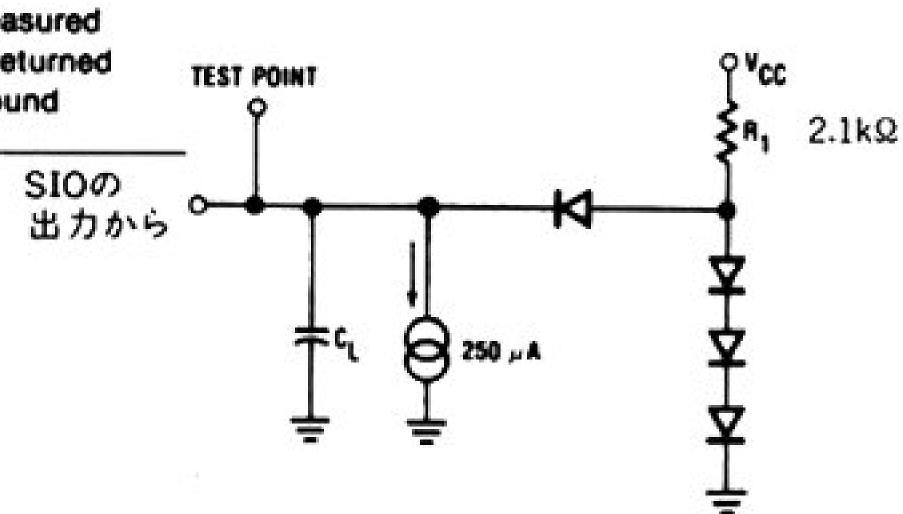
(T_a= 0°C~70°C, V_{CC}= + 5 V ± 5 %)

Symbol	Parameter	Min.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	− 0.3	+ 0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC} − 0.6	+ 5.5	V	
V_{IL}	Input Low Voltage	− 0.3	+ 0.8	V	
V_{IH}	Input High Voltage	+ 2.0	+ 5.5	V	
V_{OL}	Output Low Voltage		+ 0.4	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+ 2.4		V	I_{OH} = − 250 μA
I_{LI}	Input Leakage Current	− 10	+ 10	μA	$0 \leq V_{IN} \leq V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	− 10	+ 10	μA	$0 \leq V_{IN} \leq V_{CC}$
$I_{L(SY)}$	SYNC Pin Leakage Current	− 40	+ 10	μA	
I_{CC}	Power Supply Current		100	mA	

●容量特性●

(T_a=25°C, f= 1 MHz)

Symbol	Parameter	Min.	Max.	Unit	Test Condition
C	Clock Capacitance		40	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output Capacitance		10	pF	



C_L = 50pF. C_Lが50pF増えるごとに(最高200pF), ディレイは10ns増加する

供給されるクロックをレシーバを介して入力します．

■TxCA, TxCB：送信クロック（入力）

SIOに対する送信クロック入力です．送信データはこのクロックの立ち下がり時点に変化します．受信クロックの場合と同様に，このクロックとしてもCTCの出力をそのまま使用することができますが，同期モードまたはSDLCモードにおいてモデムとインターフェースするときには，モデムから供給されるクロックをレシーバを介して入力します．

■$\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$：リクエスト・ツー・センド（出力，"L"で有効）

この信号は通常モデムに対するRTS信号として用いられます．モデムを使用しないときなど，この信号を用いる必要のない場合には，どのような用途に用いてもかまいません．

この信号の状態はWR5のビット2の状態を反映していますが，非同期モードの場合にはWR5のビット2をリセット（"0"）しても，トランスミッタ・バッファ・エンプティになるまで，この信号は"H"になりません．

■$\overline{\text{DTRA}}$, $\overline{\text{DTRB}}$：データ・ターミナル・レディ（出力，"L"で有効）

この信号は通常モデムに対するDTR信号として用いられますが，モデムを使用しないときなど，この信号を用いる必要のない場合には，どのような用途に用いてもかまいません．この信号の状態はWR5のビット7の状態を反映しています．

■$\overline{\text{SYNCA}}$, $\overline{\text{SYNCB}}$：同期確立（入力または出力，"L"で有効）

この信号はSIOのプログラミングにより，入力として用いたり，出力として用いたりすることができます．非同期モードの場合にはこの信号は不要ですが，自動的に入力信号となり，$\overline{\text{CTS}}$や$\overline{\text{DCD}}$と同じように状態変化を外部ステータス割り込みにより知ることができ，この信号の状態はRR0のビット4に反映されます．

同期モードにおいて，外部同期モードとプログラミングした場合（WR4のビット4，5）にも，この信号は入力となります．外部同期モードの場合には，外部回路により同期の確立をはかりますが，同期が確立したときには，シンク・キャラクタの最後のビットの$\overline{\text{RxC}}$の立ち上がりから2番目の$\overline{\text{RxC}}$の立ち上がり時点に，この信号を"L"にする必要があります．

内部同期にプログラミングした場合には，この信号は出力となり，SIO内部で同期確立を検出した場合にこの信号は"L"になります．

＊

以上でSIOのピン機能の説明を終わりますが，図13にSIOのAC特性，DC特性ならびに容量特性を示します．

Z80 SIOの インターフェースと割り込み

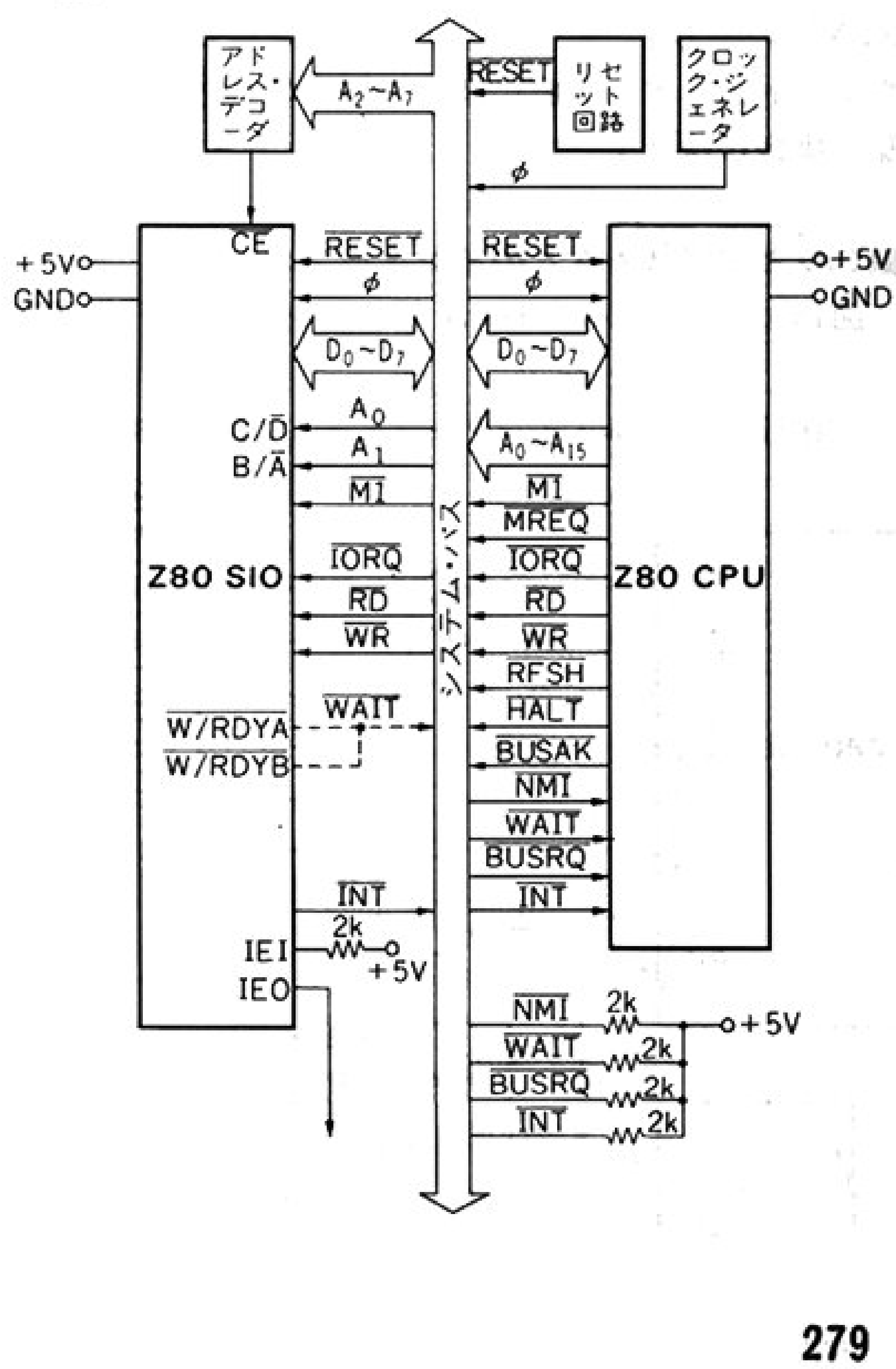
ここではZ80 SIOと-CPU，-DMAとのインターフェース，ならびにSIOとRS232Cとのインターフェース，およびZ80 SIOの割り込みについて述べます．

■Z80 SIOとZ80 CPUとのインターフェース

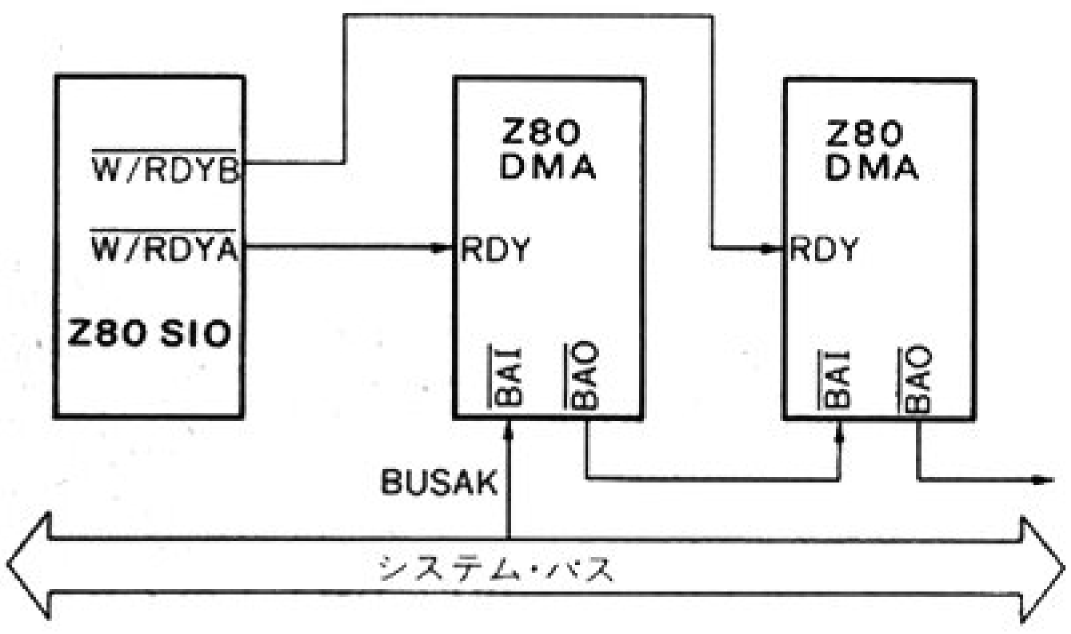
図14にSIOとZ80 CPUとのインターフェースを示します．SIOとCPUとのインターフェースは単純であり，SIOから出ているシステム・バス関係の信号をそのままCPUの信号に接続すればすみます．

この図ではSIOの割り込み優先順位を最高（IEIをプルアップ）にしてありますが，そうでない場合にはこのSIOのIEIをさらに高い優先順位を持つペリフェラルのIEOに接続します．

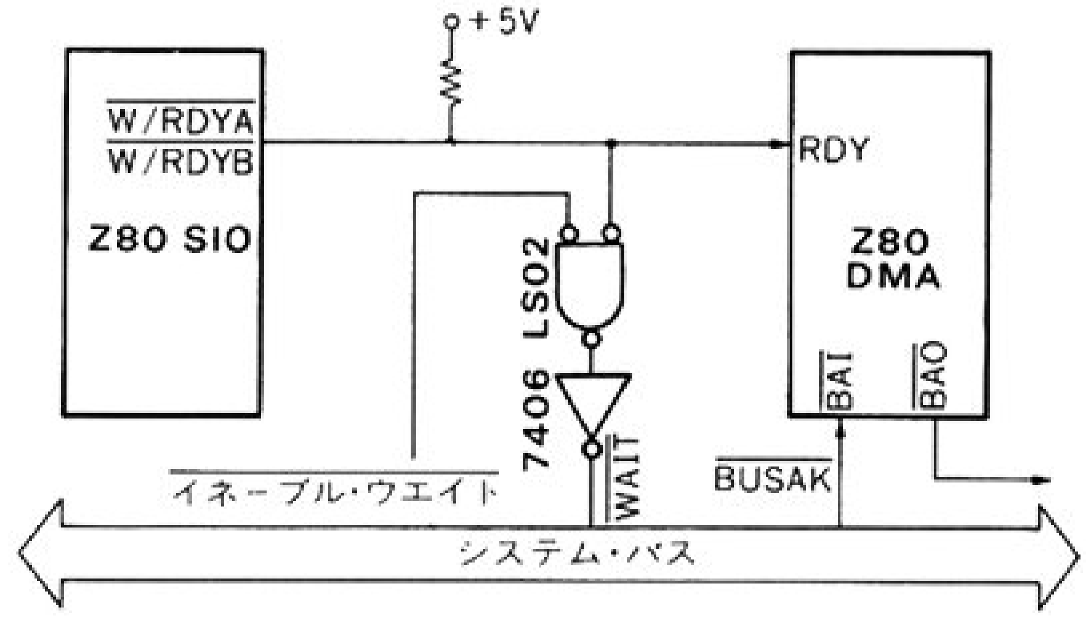
〈図14〉 Z80 SIOとZ80 CPUとのインターフェース



〈図15〉 Z80 SIO と Z80 DMA のインターフェース



〈図16〉 $\overline{WAIT}$ と RDY の混在



SIO の $\overline{W/RDYA}$ または $\overline{W/RDYB}$ を DMA の RDY とともに，CPU の $\overline{WAIT}$ としても使用する場合

SIO の $\overline{W/RDYA}$, $\overline{W/RDYB}$ は，ワイヤード接続してシステム・バスの $\overline{WAIT}$ に接続してあります（破線で示してある）．これは SIO の $\overline{W/RDYA}$ および $\overline{W/RDYB}$ を CPU に対するウェイト信号として使用する場合であり，SIO のウェイト機能を用いない場合には不要です．

$\overline{W/RDYA}$, または $\overline{W/RDYB}$ をウェイト機能として用いる場合には，SIO をそれなりにプログラミングすること（WR0）が必要であり，これらの信号をまちがってもレディ機能としてプログラミングしてはなりません．

SIO はリセットされると，$\overline{W/RDYA}$ および $\overline{W/RDYB}$ の出力を，オープン・ドレインにしますので，ウェイト／レディ機能に関して何もプログラミングしない場合には無事です．

■SIO と Z80 DMA とのインターフェース

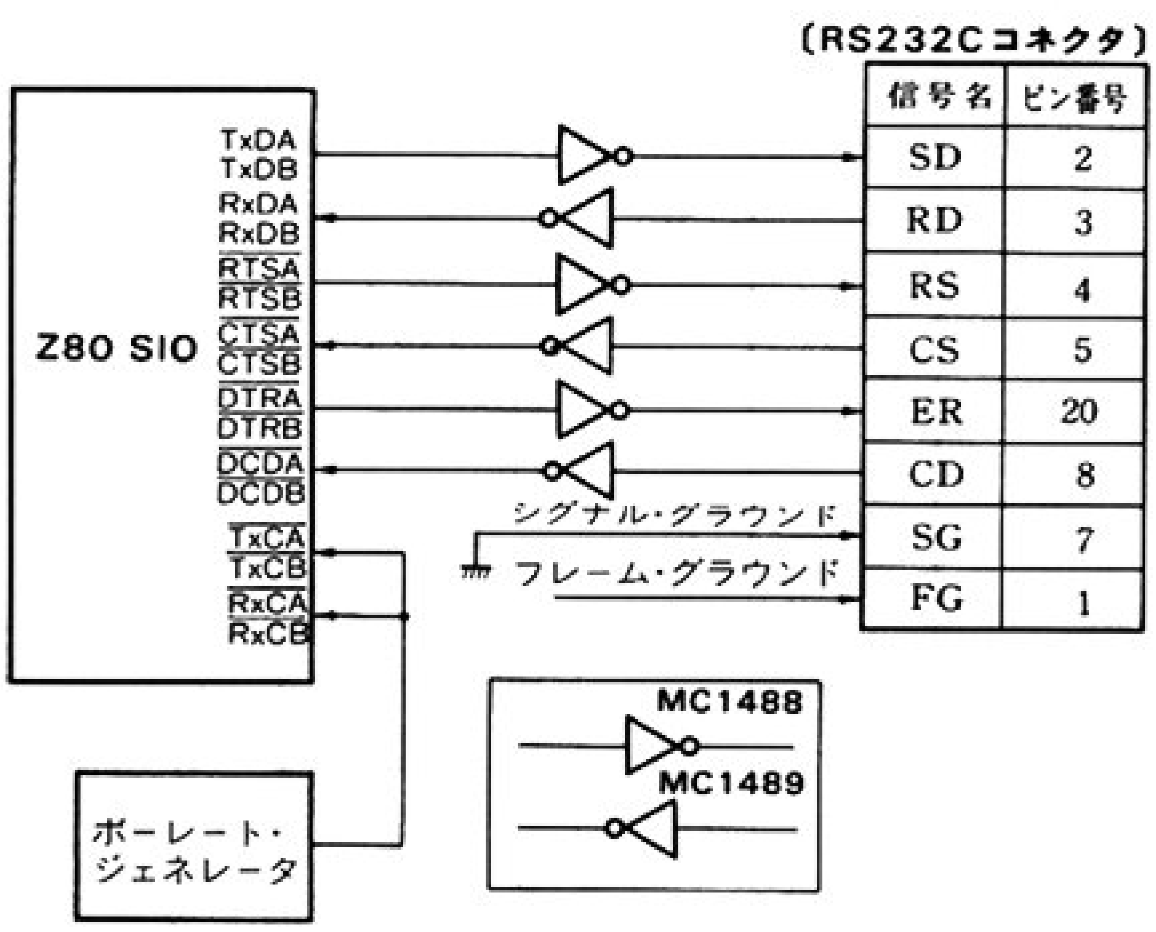
図15に SIO と Z80 DMA とのインターフェースを示します．この場合 SIO の $\overline{W/RDYA}$ または $\overline{W/RDYB}$ は，レディ機能として動作するようにプログラミングする必要があります．また SIO にプログラミングするレディの有効極性と，DMA にプログラミングするレディの有効極性はマッチさせる必要のあることはいうまでもありません．

図15では SIO のチャネル A とチャネル B が同時に DMA データ転送が可能なように，2個の DMA を用いていますが，両チャネルが同時に DMA データ転送を行わない場合には DMA を1個だけにし，$\overline{W/RDYA}$ と $\overline{W/RDYB}$ をマルチプレックスして DMA の RDY に接続し，外部回路によりどちらのレディ（$\overline{W/RDYA}$ か $\overline{W/RDYB}$）を使用するかの指定を行うようにします．

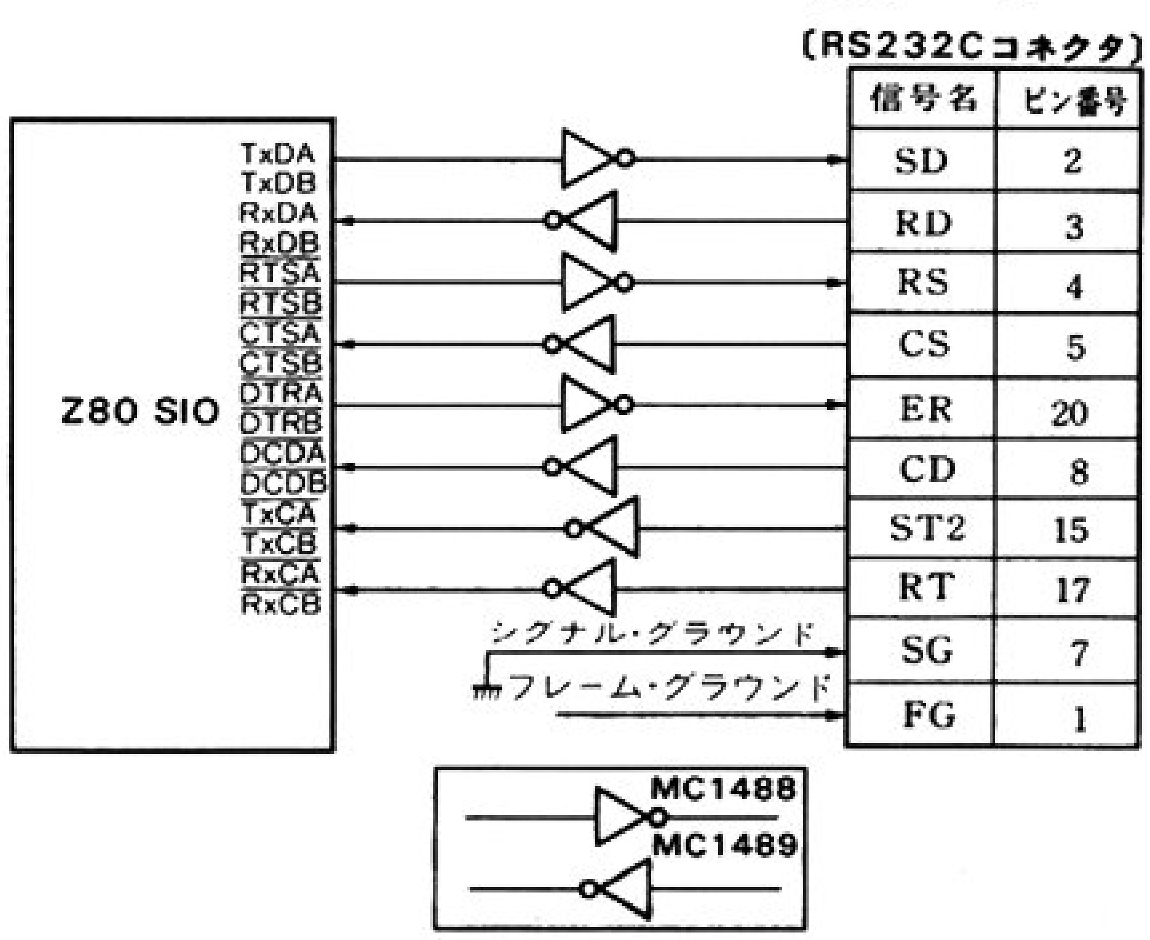
図16に $\overline{W/RDYA}$, $\overline{W/RDYB}$ をあるときにはウェイト機能として用い，あるときにはレディ機能として用いるような場合の構成を示します．

この場合にはイネーブル・ウェイトという信号を I/O ポートなどで作成し，この信号と $\overline{W/RDYA}$, または

〈図17〉 モデム・インターフェース（非同期モード）



〈図18〉 モデム・インターフェース（同期モード）



$\overline{\text{W/RDYB}}$ を AND ゲートを介した後に，オープン・コレクタのドライバ(図16では7406)により，$\overline{\text{WAIT}}$ ラインを駆動します．

■SIO とモデムとのインターフェース

SIO はシリアル・インターフェースであるので，コミュニケーション・ラインを通じて他の装置と接続されますが，ほとんどの場合 RS232C モデム・インターフェースが使われます．ここでは非同期モードおよび同期モードにおける，SIO とモデムとのインターフェースについて説明をします．

図17に非同期モードにおけるモデム・インターフェースを示します．SIO の各信号は RS232C ドライバ／レシーバ（この図では MC1488/MC1489）を通じて，RS232C コネクタに接続されます．非同期モードでは SIO 送信クロックと受信クロックは内部のボーレート・ジェネレータから供給されます．

図18に同期モードにおけるモデム・インターフェースを示します．同期モードでは送信クロックと受信クロックはモデムから供給されます．送信クロックは，ST2 (RS232C コネクタのピン⑮) から供給され，受信クロックは RT (同コネクタのピン⑰) から供給されます．

RS232C インターフェースを持った装置はモデムと接続されるのが本来ですが，場合によっては RS232C インターフェースを持った装置同士をモデムを介さずに直接接続する必要があります．

両装置，たとえば CRT ターミナルとパーソナル・コンピュータが近距離にある場合には，高価なモデムを使用しなくても十分コミュニケーションができるはずだからです．

このように RS232C インターフェースを持った装置をモデムを介さずに直接接続することをダイレクト・コネクションと呼びます．

この場合，接続される両装置はモデムを相手とするインターフェースを持っていますので，両装置の RS232C コネクタを一対一で接続したのではダメで，両コネクタを結ぶケーブル内で信号線をひねってやる必要があります．

図19に非同期モードにおけるダイレクト・コネクションの方法を示します．SD (送信データ) は互いに相手の RD (受信データ) と接続するようにひねり，RS (リクエスト・ツー・センド) は相手の CD (キャリヤ・デテクト)，ER (データ・ターミナル・レディ) は相手の CS (クリヤ・ツー・センド) と接続するようにひねります．

このような接続をすることにより，両装置はあたかもモデムと信号のやりとりをしているかのようにごまかされて（しかし正常に）動作します．

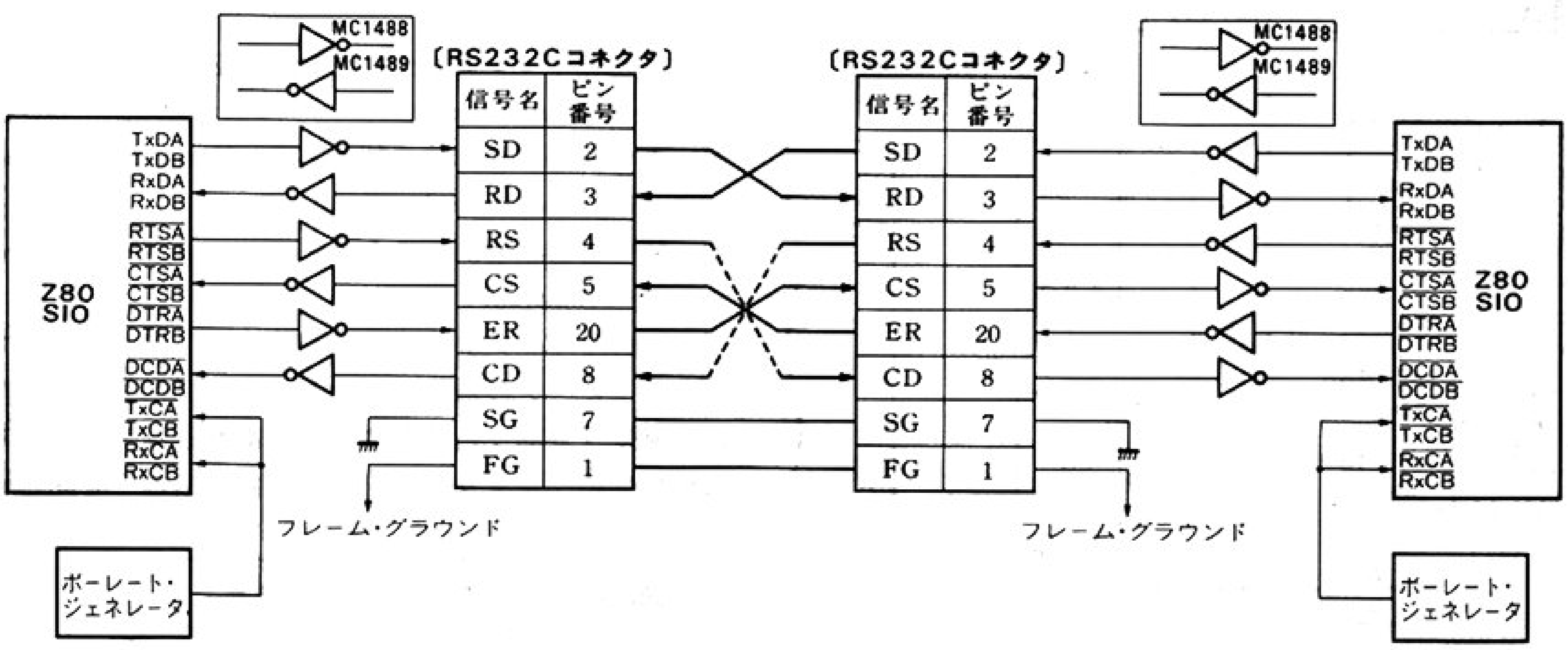
図20に同期モードにおけるダイレクト・コネクションの方法を示します．同期モードでは送信クロックと受信クロックをモデムからもらうのですが，モデムがありませんので互いに相手に対して受信クロックを供給する必要があります．

同期モードにおけるダイレクト・コネクションでは，内部にボーレート・ジェネレータが必要となり，その出力を SIO の送信クロックとすると同時に，ドライバを介して，RS232C コネクタの ST1 (ピン㉔) に出力します．そして ST1 と RT をひねることにより互いに受信クロックを供給しあうことができます．

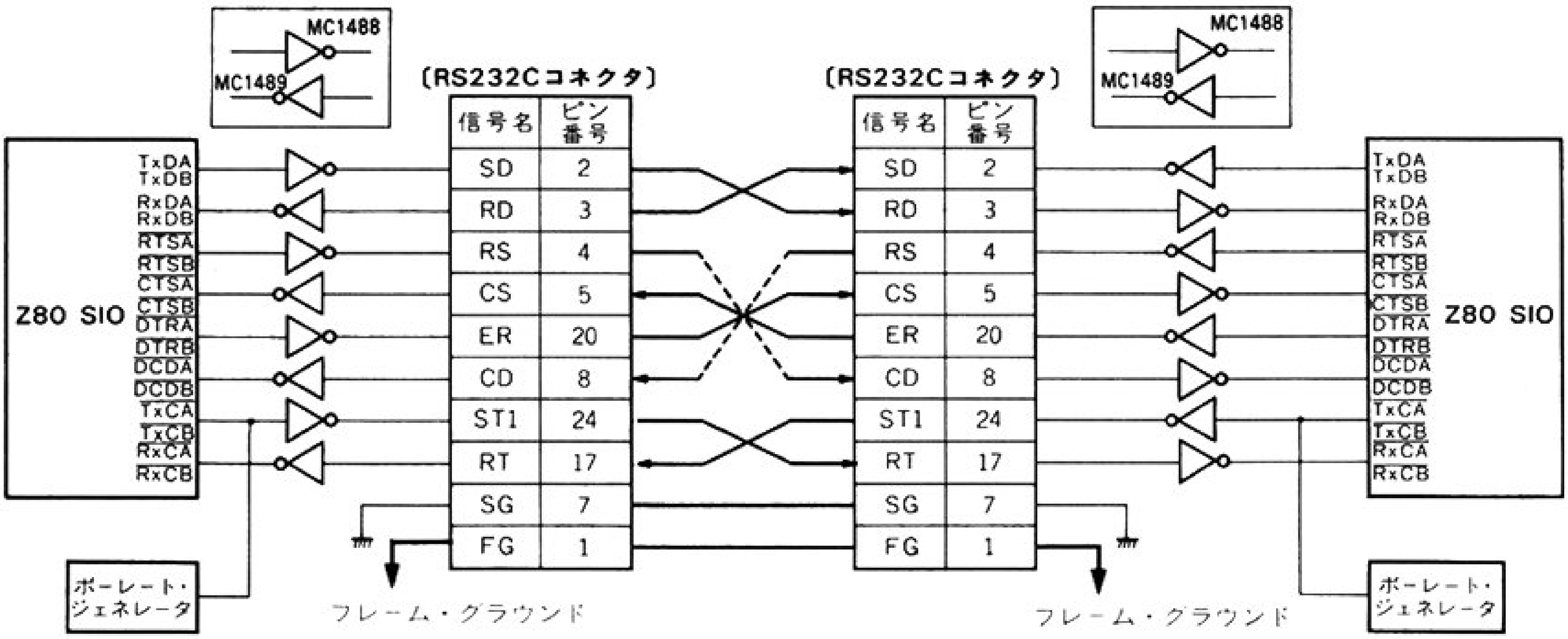
＊

以上の説明からおわかりいただけたと思いますが，非同期モード，同期モード，モデム接続，ダイレクト・

〈図19〉 ダイレクト・コネクション（非同期モード）



〈図20〉 ダイレクト・コネクション（同期モード）



コネクション，これらの中で異なるのはSIOに供給する送・受信クロックだけです．

したがって送受信クロックが切り替えられるようなくふうをしておけば，上記のいずれでも動作が可能ということになります．このクロック切り替えを行うロジック（クロック・マルチプレクサ）を図21に示します．

このロジックはA，B両入力信号により，マルチプレクサ（LS153）を切り替え，必要な送信クロックと受信クロックを選択して出力します．

Z80 SIOの割り込み

Z80 SIOも他のペリフェラル，たとえば-PIO，-CTC，-DMAなどと同様に，ディジー・チェイン式の割り込み優先順位制御機能，ならびに割り込みベクトル発生機能を持っていますが，このほかにSIOは“ステータス・アフェクツ・ベクトル（Status Affects Vector）”という機能を備えています．

このステータス・アフェクツ・ベクトルというのは，「SIOの状態により割り込みベクトルが変化する」という機能であって，割り込み要因に応じてSIOが割り込みベクトルを4種類出してくれるということです．

したがって，割り込みルーチンの入口点を要因ごとに設けることができますので，割り込み要因解析に要するオーバヘッドを減じることができます．

図22にSIOの割り込み要因の構成を示します．SIOの割り込み要因は以下の三つに分けられます．すなわち，

❶ 受信割り込み（特殊受信状態割り込みを含む）
❷ 送信割り込み
❸ 外部／ステータス割り込み

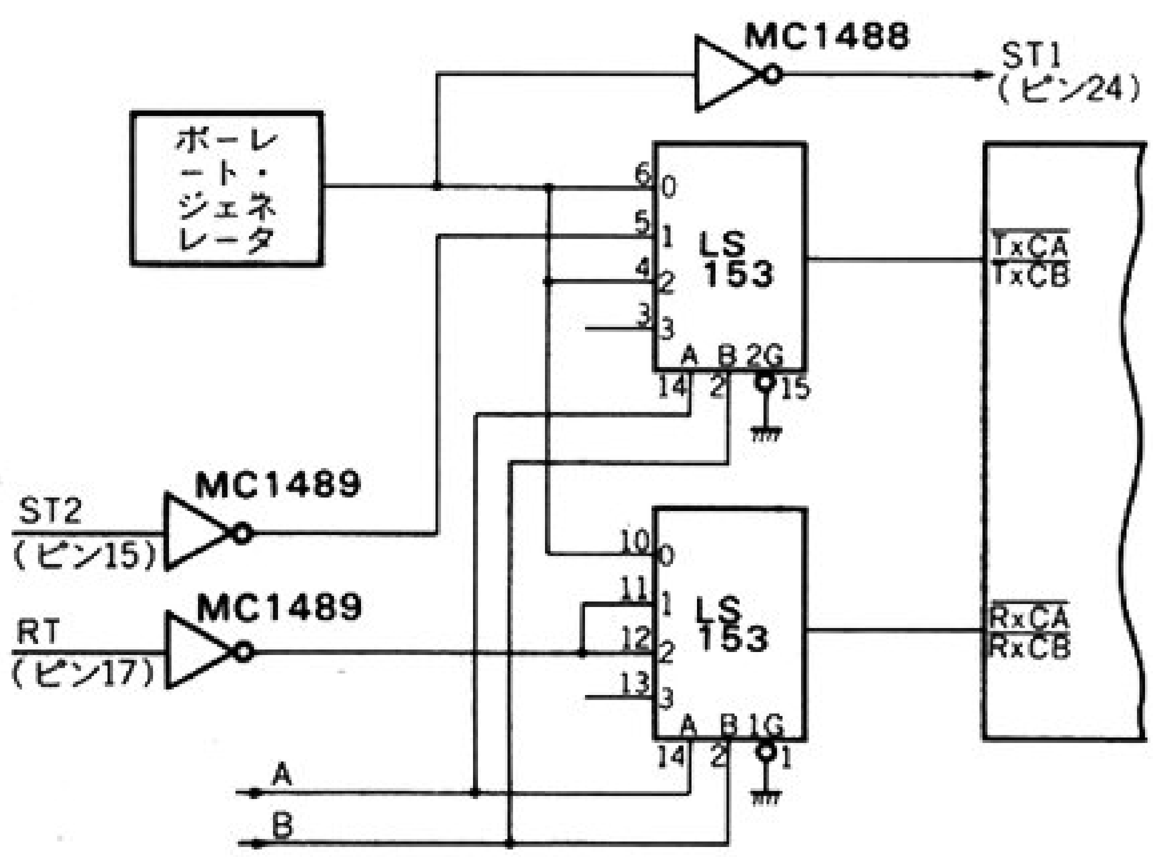
割り込み優先順位はチャネルA，チャネルBの順であり，各チャネル内での優先順位は❶～❸の順です．

受信割り込み要因はさらに次の3種類に分けられます．

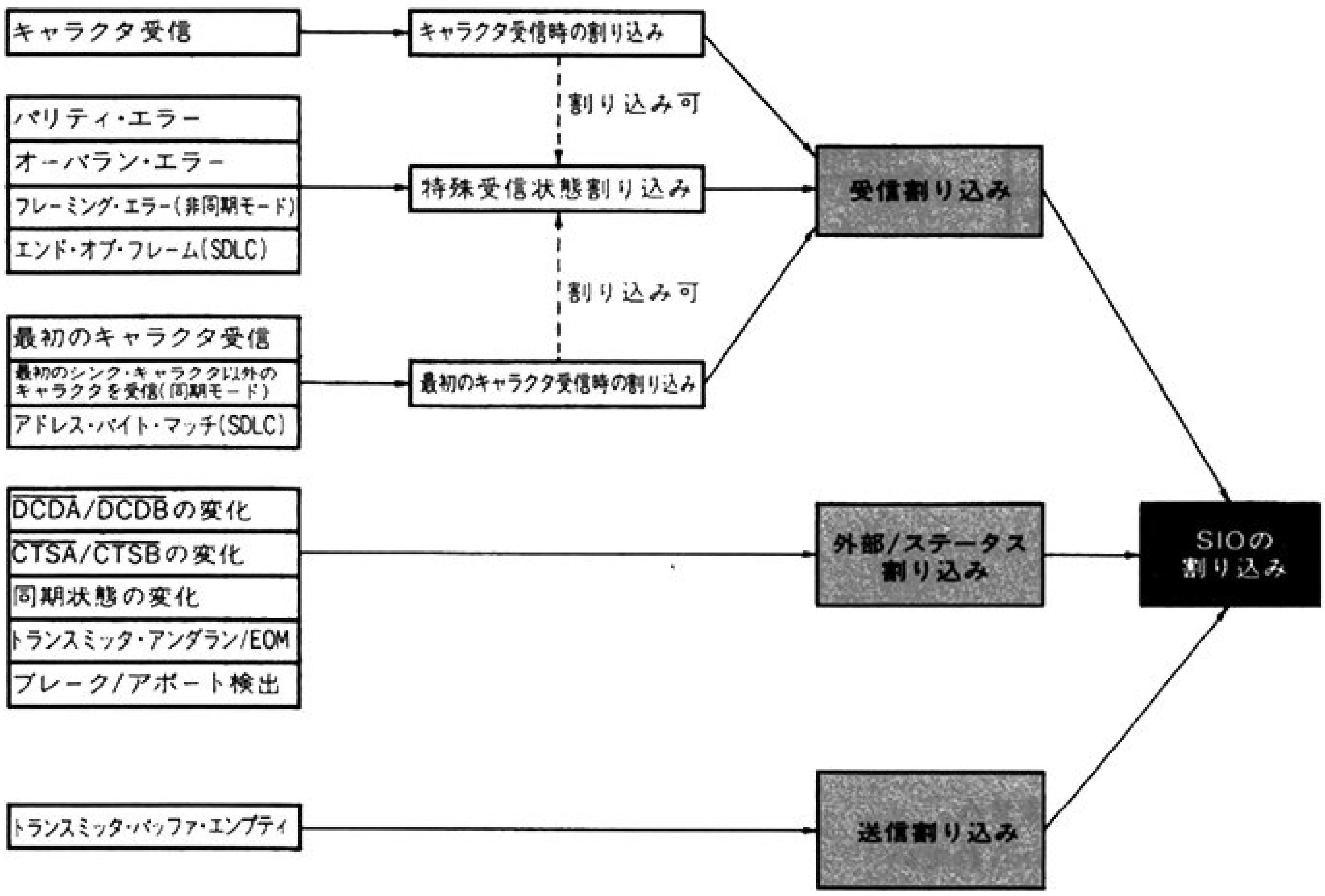
ⓐ 最初のキャラクタ受信時の割り込み（Interrupt on first receive character）
ⓑ キャラクタ受信時の割り込み （Interrupt on all receive characters）
ⓒ 特殊受信状態割り込み（Interrupt on a special receive condition）

“最初のキャラクタ受信時の割り込み”は文字通りブロック・データの最初のキャラクタを受信したときだけに発生する割り込みです．この割り込みが発生する状態はⓑの“キャラクタ受信時の割り込み”と同じで

モード ＼ クロック	送信クロック	受信クロック	A	B
非同期モード	内部クロック	内部クロック	L	L
同期モード（モデム）	ST2	RT	H	L
同期モード（ダイレクト）	内部クロック	RT	L	H



〈図21〉 クロック・マルチプレクサ



<図22>
Z80 SIOの割り込み構成

すが，1回割り込みを発生すると再度割り込みがイネーブルされるまで，データを受信しても割り込みを発生しません．

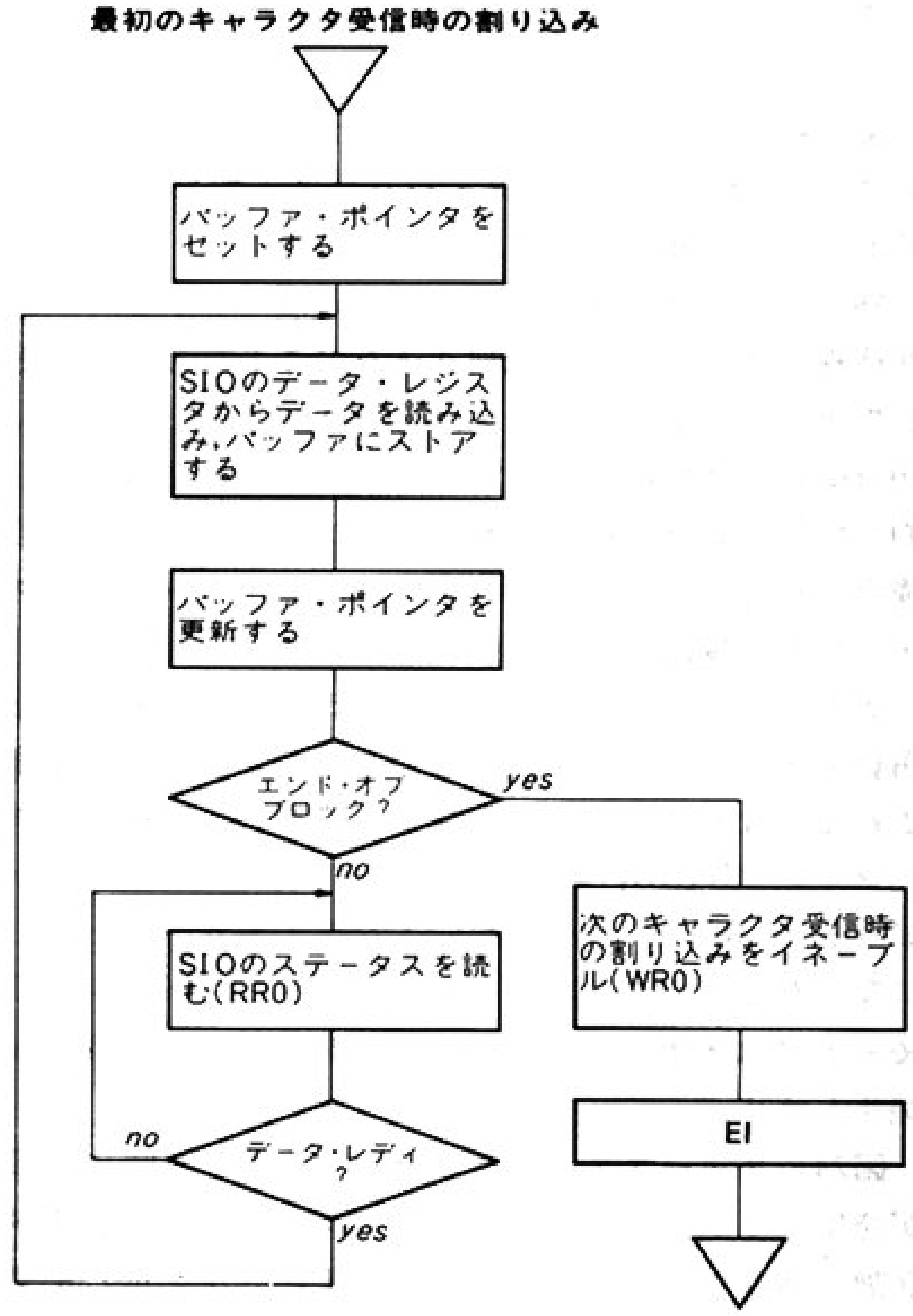
この割り込みはいつ入ってくるかわからないような高速データ・ブロックの受信に使うことができます．データ・ブロックの最初のデータがレシーバに到達すると，この割り込みが発生します．

割り込みルーチンでは最初のデータを読み込んだ後，プログラムの制御を他に渡すようなことはせず，割り込み禁止のままポールド・モードのデータ転送を行うことにより，SIOからのデータをバッファにためていきます．

データ・ブロックの受信が終わると，プログラムの制御を元に戻し，バッファ内のデータを処理します．割り込みルーチンがプログラムの制御を元に戻す前に，“次のキャラクタ受信時の割り込みをイネーブル”というコマンドを出しておくことにより(WR0)，次のデータ・ブロックの受信開始時に再度割り込みが発生します．以上の説明を図23にフロー・チャートにして示します．

“キャラクタ受信時の割り込み”は，受信したすべてのキャラクタに対して発生する割り込みであり，ふつうの受信割り込みです．“特殊受信状態割り込み”は，受信したデータにパリティ・エラー，オーバラン・エラーやフレーミング・エラーを検出したときに発生します．ただしパリティ・エラーをこの割り込みに含めるかどうかは，プログラミングにより指定することができます．また，SDLCではエンド・オブ・フレーム(クロージング・フラグ)を検出したときに発生します．

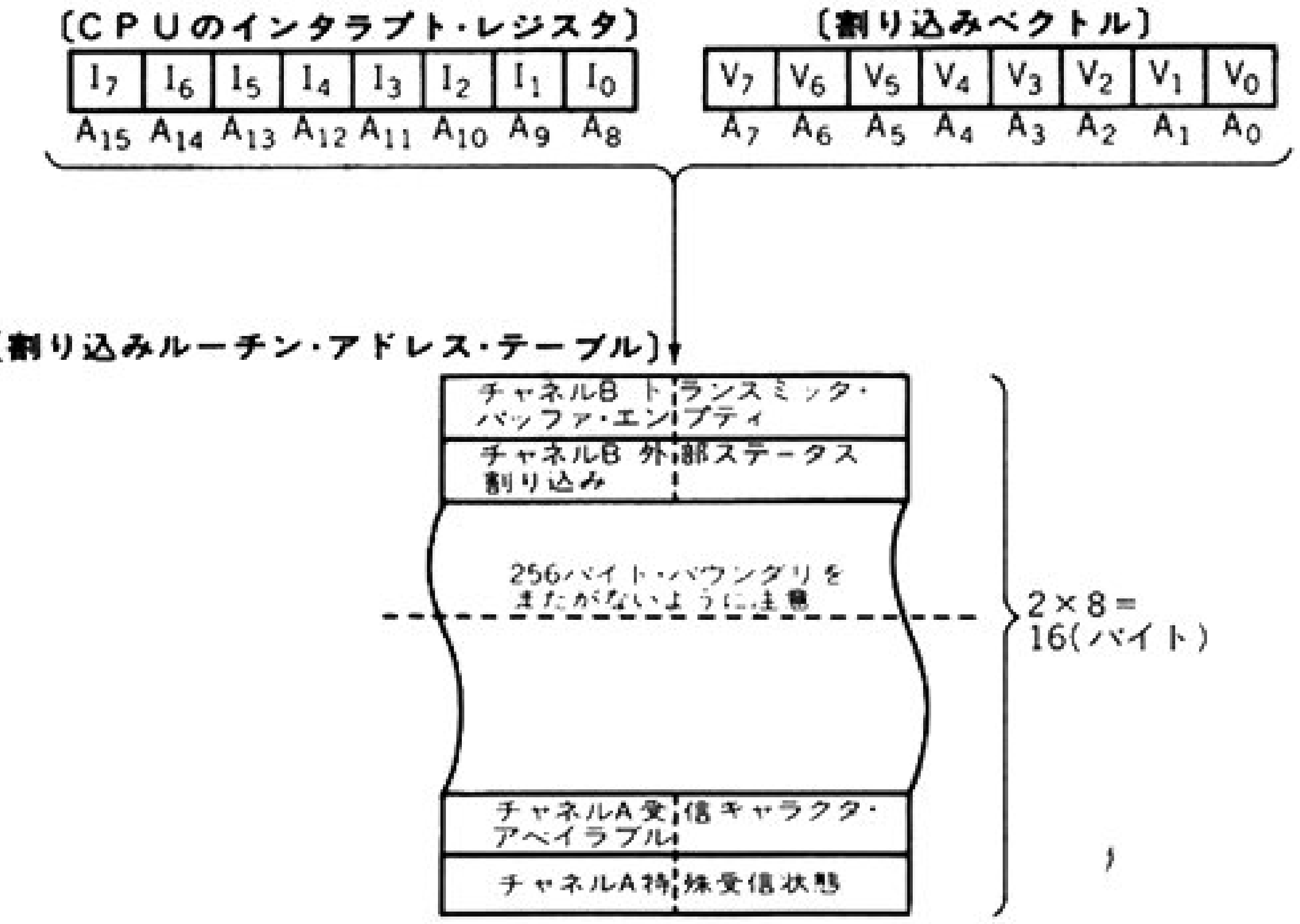
<図23>　最初のキャラクタ受信時の割り込みを用いたデータの受信



この割り込みはレシーバ割り込み（キャラクタ受信時の割り込み，または最初のキャラクタ受信時の割り

〈図24〉 ステータス・アフェクツ・ベクトル時の割り込みベクトルと割り込みルーチン・アドレス・テーブル

	ベクトルのビット1～3			
	V_3	V_2	V_1	割り込み要因
チャネルB	0	0	0	チャネルBトランスミッタ・バッファ・エンプティ
	0	0	1	チャネルB外部/ステータス割り込み
	0	1	0	チャネルB受信キャラクタ・アベイラブル (Available)
	0	1	1	チャネルB特殊受信状態
チャネルA	1	0	0	チャネルAトランスミッタ・バッファ・エンプティ
	1	0	1	チャネルA外部/ステータス割り込み
	1	1	0	チャネルA受信キャラクタ・アベイラブル (Available)
	1	1	1	チャネルA特殊受信状態



込み）がイネーブルされていないと発生しません．

最初のキャラクタ受信時の割り込みが用いられている場合には，パリティ・エラー以外の特殊受信状態が発生すれば，最初のキャラクタ受信の後（第２番目以降のデータ受信）であっても，この割り込みは発生します．

送信割り込みは単純であり，トランスミッタのバッファが空（トランスミッタ・バッファ・エンプティ）のときにだけ発生します．

外部／ステータス割り込みは，正確にいうと外部割り込みとステータス変化割り込みのことであり，主としてモデム制御用のライン（$\overline{CTSA}$/$\overline{CTSB}$，$\overline{DCDA}$，$\overline{DCDB}$）の状態の変化をプログラムに知らせるのが目的ですが，同期モードにおけるシンク・キャラクタの検出，および SDLC モードにおける，オープニング・フラグの検出時にも発生します．

また，トランスミッタ・アンダラン状態（データを送信するタイミングに，送信バッファにデータが用意されていなかったとき）が発生した場合，受信データ・ライン（RxDA，RxDB）がブレーク状態（オール・ビット＝"0"で，かつフレーミング・エラーのとき）になったときにもこの割り込みが発生し，かつ SDLC モードではアボート・シーケンスを受信した場合にも発生します．

図24にステータス・アフェクツ・ベクトル時に SIO が発生する割り込みベクトルとプログラムが用意すべき割り込みルーチンのアドレス・テーブルを示します．

ステータス・アフェクツ・ベクトルと指定されると，SIO はプログラミングされた割り込みベクトル（チャネルBのWR2）のビット１～３を割り込みに関連したチャネルと割り込み要因にしたがって変化させます．

割り込みベクトルは全部で８個あり，アドレス・テーブルは各ベクトルに対して２バイト必要ですので，合計16バイトとなります．

Z80 CPU は割り込みベクトルから，割り込みルーチンのアドレス・テーブルをアクセスする際，CPU のインタラプト・レジスタの内容をテーブル・アドレスの上位８ビットとして用います．

したがって割り込みルーチンのアドレス・テーブルが，メモリの 256 バイト境界をまたがないように注意をはらう必要があります．

Z80 SIO の コントロール・レジスタ

SIO を動作させるには，SIO のコントロール・レジスタに対して適当な値をセットしたり，コントロール・レジスタからステータスを読み込んで，何らかの判断を行う必要があります．

SIO のコントロール・レジスタはライト・レジスタとリード・レジスタからなります．ライト・レジスタは８個あり，WR0～WR7 という名が付けられています．このライト・レジスタには SIO をコントロールするためのコマンドを書き込みます．

また，リード・レジスタは３個あり，RR0～RR2 という名が付けられています．このリード・レジスタには SIO のステータスがセットされ，プログラムはこれを読んで SIO の状態を知ります．

ここでは SIO のライト・レジスタとリード・レジスタについて詳しく説明をします．

ライト・レジスタおよびリード・レジスタの構成

図25にSIOの8個のライト・レジスタ，およびリード・レジスタの構成を示します．ライト・レジスタ0，1，3～7はチャネルA，チャネルBそれぞれ独立に持っていますが，ライト・レジスタ2だけは例外でチャネルBにしかありません．

ライト・レジスタ（以下WR）0はSIOに対するコマンドおよびポインタ・ビットからなります．WR1は主として割り込み関係のコントロールをしますが，$\overline{WAIT}$/READYのコントロールもします．

WR2は前述のごとくチャネルBにしかありませんが，これには割り込みベクトルがストアされます．WR3はレシーバ関係のコントロールをし，WR4はレシーバおよびトランスミッタに共通のコミュニケーション・モードを管理します．

WR5はトランスミッタのコントロールに用いられ，WR6およびWR7は同期モードにおけるシンク・キャラクタとSDLCモードにおけるフラグ・パターンと従局アドレスを保持します．

リード・レジスタ（以下RR）0は割り込み要因の識別に用いられますが，ポールド・モードでSIOを動かすときにはトランスミッタとレシーバの状態を知るのに用いられます．

RR1は特殊受信状態割り込みの要因を知るのに用いられ，エラー状態およびエンド・オブ・フレーム状態，(SDLC) を示しており，かつSDLCモードにおけるレシデュー（Residue）・コードをプログラムに伝えるのに用いられます．RR2は割り込みベクトルを保持しています．

ライト・レジスタおよびリード・レジスタへのアクセス方法

ライト・レジスタ，およびリード・レジスタに対するアクセスは，SIOのピン$C/\overline{D}$を"H"にして行われます．

SIOはライト・レジスタ，およびリード・レジスタにアクセスするためのポインタ（WR0のビット0～2）を持っています．このポインタは次にリード／ライトされるレジスタ番号（RR0～2，WR0～7）を示します．SIOはリセットされると，ポインタを"000"にします．

したがって，リセット後にSIOに対してリードまたはライト命令がくると，それはRR0またはWR0に対する命令であるとみなされます．

WR1～WR7，およびRR1～RR2をアクセスする場合には，まずWR0に対してポインタを書き込みます．これによりSIOは次にリードまたはライトされる対象はこのポインタで示されるレジスタであると解釈します．

SIOはWR1～WR7またはRR1～RR2に対するアクセスが終了するとポインタをゼロ・クリヤします．したがって，その次にアクセスされる対象はWR0またはRR0となります．そして新たにポインタをWR0に書き込むことにより，他のレジスタをアクセスすることができます．

図26にライト・レジスタ，およびリード・レジスタにアクセスする方法を示します．

〈図25〉 Z80 SIOのライト・レジスタ，およびリード・レジスタの構成

〔ライト・レジスタの構成〕

レジスタ名	記号	役　目
ライト・レジスタ0	WR0	コマンドおよびポインタ・ビット
ライト・レジスタ1	WR1	割り込み制御および$\overline{WAIT}$/READYの制御
ライト・レジスタ2	WR2	割り込みベクトル（チャネルBのみ）
ライト・レジスタ3	WR3	レシーバ制御
ライト・レジスタ4	WR4	コミュニケーション・モードの設定
ライト・レジスタ5	WR5	トランスミッタ制御
ライト・レジスタ6	WR6	シンク・キャラクタまたは従局アドレス(SDLC)
ライト・レジスタ7	WR7	シンク・キャラクタまたはフラグ・パターン(SDLC)

〔リード・レジスタの構成〕

レジスタ名	記号	役　目
リード・レジスタ0	RR0	割り込み要因の識別
リード・レジスタ1	RR1	エラー状態の識別およびレシデュー・コード
リード・レジスタ2	RR2	割り込みベクトル（チャネルBのみ）

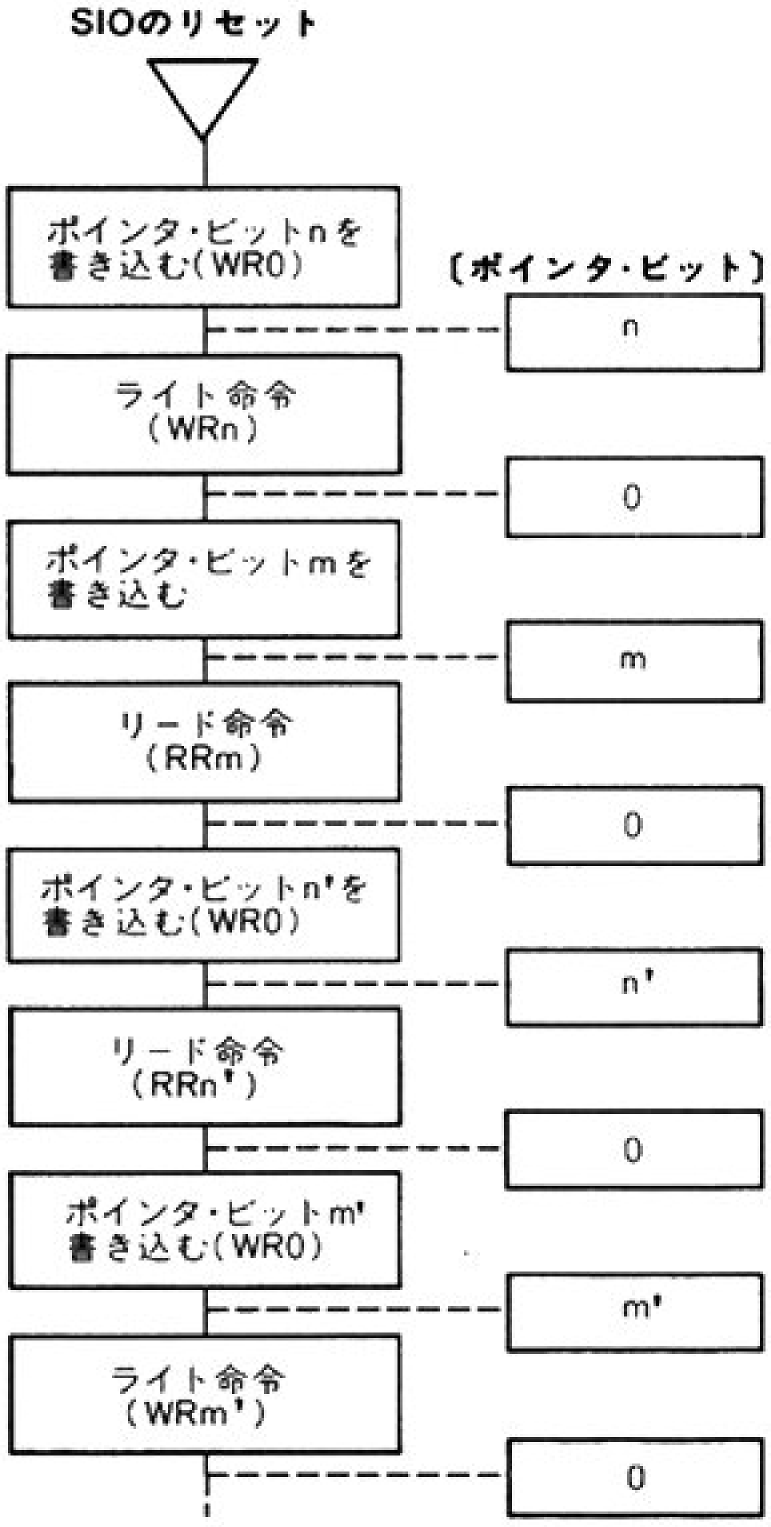
ライト・レジスタ

ライト・レジスタ0（WR0）

図27にライト・レジスタ0（WR0）の構成を示します．WR0は上位2ビット（D_7，D_6）がCRCチェッカ，およびトランスミッタ・アンダーラン・ラッチの制御に用いられ，下位3ビットはポインタとして使用されています．

またこのWR0のビット3～5は組み合わせにより，8種類のコマンドとなります．WR0のビット3～7によりSIOに対して必要な指示をするのと同時に，ビット0～2により，次にアクセスすべきレジスタ番号（ポインタ・ビット）を指定することができます．

〈図26〉 ライト・レジスタ，およびリード・レジスタに対するアクセス方法



〈図27〉 ライト・レジスタ0(WR0)の構成

D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0

ポインタ（D_2 D_1 D_0）

D_2	D_1	D_0	
0	0	0	レジスタ0
0	0	1	レジスタ1
0	1	0	レジスタ2
0	1	1	レジスタ3
1	0	0	レジスタ4
1	0	1	レジスタ5
1	1	0	レジスタ6
1	1	1	レジスタ7

コマンド（D_5 D_4 D_3）

D_5	D_4	D_3	コマンド番号	
0	0	0	0	ノー・オペレーション
0	0	1	1	センド・アボート(SDLC)
0	1	0	2	外部/ステータス割り込みのリセット
0	1	1	3	チャネル・リセット
1	0	0	4	次のキャラクタ受信時の割り込みを可とする
1	0	1	5	保留中の送信割り込みをリセットする
1	1	0	6	エラー・リセット
1	1	1	7	RETI(チャネルAのみ)

D_7	D_6	
0	0	ノー・オペレーション
0	1	受信CRCチェッカのリセット
1	0	送信CRCジェネレータのリセット
1	1	送信アンダラン/EOMラッチのリセット

■ビット0〜2(ポインタ)

これらのビットはポインタであり，次にリードまたはライトするレジスタ番号を指定します．

■ビット3〜5(コマンド)

これら3ビットは8種類のコマンドの意味を持ちます．ビット3〜5の組み合わせにより，コマンド0〜コマンド7という名前が付けられています．

・コマンド0(ノー・オペレーション)

このコマンドはノー・オペレーションであり，これが出されてもSIOは何もしません．

このコマンドの目的はSIOの他の状態を変えずに，ポインタだけをセットすることにありますが，一連のSIOに対するコマンド列（コマンド・チェイン）中の特定のコマンドを無効にしたり，あとから挿入されるかもしれないコマンドのためにコマンド・チェインの場所をリザーブしておくのにも用いることができます．

・コマンド1(センド・アボート)

このコマンドは“センド・アボート（Send Abort)”と呼ばれ，SDLCモードにおいてアボート・シーケンス（7個以上の連続した“1”）を発生させるのに用いられます．

・コマンド2(外部ステータス割り込みのリセット)

このコマンドは外部／ステータス割り込みをリセットするのに用いられます．$\overline{CTSA}/\overline{CTSB}$ や $\overline{DCDA}/\overline{DCDB}$ 等の状態が変化すると，SIOは変化時点の状態をRR0にラッチし，外部/ステータス割り込みを発生しますが，このコマンドが出されるまで次の状態変化を無視します．

したがって外部／ステータス割り込みが発生した後，次の状態変化が生じたときに外部／ステータス割り込みを起こしたい場合には，このコマンドを出す必要があります．

また，外部／ステータス割り込みルーチン内で，割り込みが発生した時点ではなく，その時点の状態を知りたいときには，このコマンドを出した直後にRR0を読み込みます．

・コマンド3(チャネル・リセット)

このコマンドはチャネルをリセットする機能を持ちます．このコマンドの働きはSIOのピンに出ている $\overline{RESET}$ 信号の機能と同じですが，リセットされる対象がこのコマンドが出されたチャネルに対してのみである点が異なります．

チャネルAに対してこのコマンドが出されると，割

り込み制御ロジックもリセットされます．このコマンドを出したチャネルに対しては，ライト・レジスタの値をすべて書き直す必要があります．

・コマンド4
（次のキャラクタ受信時の割り込みをイネーブルする）

このコマンドは“次のキャラクタ受信時の割り込み”をイネーブルします．つまり“最初のキャラクタ受信時の割り込み”を用いてSIOからデータ・ブロックの受信を行い，エンド・オブ・ブロックを検出した後，次のデータ・ブロックの先頭データを受信したときに割り込みを発生させたいとき，このコマンドを使用します．

・コマンド5（保留中の送信割り込みのリセット）

このコマンドは，保留中の送信割り込み（トランスミッタ・バッファ・エンプティ）をリセットします．このコマンドは送信割り込みが発生したけれども，データを全部送信してしまったので，送るべきデータがもうないような状態のときに使います．

・コマンド6（エラー・リセット）

このコマンドはエラー・ステータス（パリティ・エラー，オーバラン・エラー）をリセットするのに用いられます．パリティ・エラー，およびオーバラン・エラーが発生すると，SIOはこれらの状態をRR1にラッチします．

したがって，これらの状態はいったん発生するとこのコマンドが出されるまで変化しませんので，データを全部受信し終わった時点にチェックすることもできます．チェックが終わったら，このコマンドを出してエラーをクリヤします．

このコマンドにより，フレーミング・エラー（RR1のビット6）もリセットされますが，フレーミング・エラー・ステータスは，キャラクタの受信ごとに更新されますので，このコマンドを出してクリヤする必要はありません．

・コマンド7（RETI）

Z80ファミリのペリフェラルは，Z80 CPUがRETI命令（$ED4D_{16}$）を実行するのを監視しており，この命令が実行されると割り込みディジー・チェインの制御を行います．このコマンドを受け取ると，SIOはCPUがRETI命令を実行したのを検出したのと同じ動作をします．

このコマンドはZ80 CPU以外のCPUとSIOをインターフェースするために設けられています．なお，このコマンドはチャネルAに対してのみ有効です（$B/\overline{A}$＝“L”）．

■ビット6，7

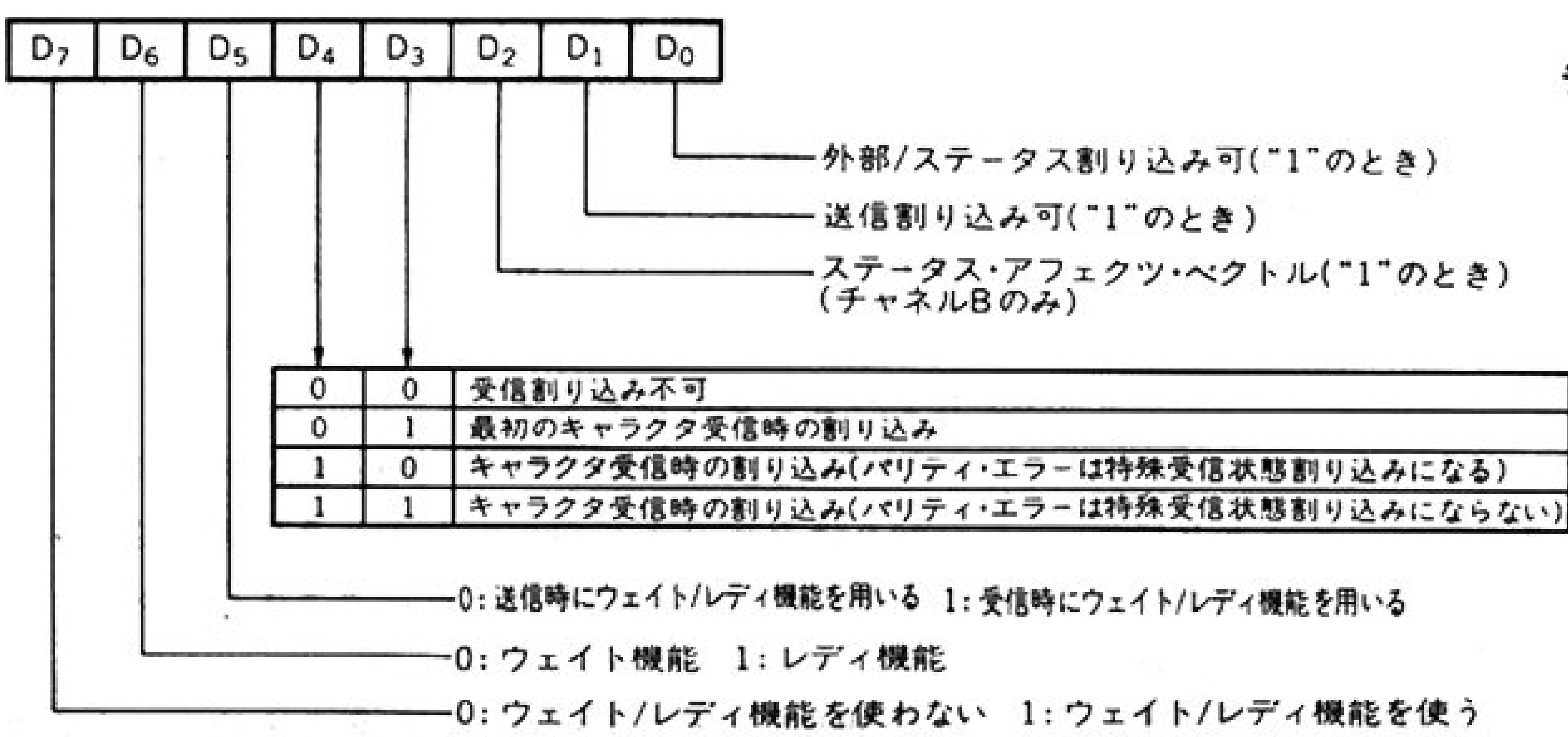
ライト・レジスタ0のビット6と7はCRCチェッカ，およびCRCジェネレータのリセット，ならびにトランスミッタ・アンダラン/EOMラッチのクリヤに用いられます．

ビット6，7が“00”の場合はノー・オペレーションであり，CRCチェッカ／ジェネレータ，およびトランスミッタ・アンダラン/EOMラッチの状態には何の影響もおよぼしません．

ビット6，7が“10”であるとCRCチェッカがリセットされ，ビット6，7が“01”であるとCRCジェネレータがリセットされます．同期モードの場合にはCRCチェッカ／ジェネレータはともにゼロ・クリヤされますが，SDLCモードでは“オール1”にリセットされます．ビット6，7が“11”であるとトランスミッタ・アンダラン/EOMラッチがクリヤされます．

ライト・レジスタ1（WR1）

ライト・レジスタ1は割り込み関係の制御および$\overline{W/RDYA}$，$\overline{W/RDYB}$ピンの使用法（ウェイト／レディ機能）を指定します．図28にライト・レジスタ1の構成を示します．



〈図28〉
ライト・レジスタ1（WR1）の構成

■ビット0(外部/ステータス割り込みの可否)

このビットが“0”であると外部/ステータス割り込みは禁止され,“1”であると可となります.

■ビット1(送信割り込みの可否)

このビットが“0”であると,トランスミッタ・バッファ・エンプティ時の割り込みが禁止され,“1”であると可となります.

■ビット2(ステータス・アフェクツ・ベクトル)

このビットが“1”であるとステータス・アフェクツ・ベクトルとなり,割り込み要因に応じて割り込みベクトルのビット1~3が変化します.このビットが“0”であると割り込み要因が何であっても同一の割り込みベクトルが出されます.なお,このビットはチャネルBのWR1に対してのみ有効です.

■ビット3,4(受信割り込みの指定)

これら2ビットの組み合わせにより,受信割り込みの指定を行います.ビット3,4が“00”であると,受信割り込みは禁止されます.

ビット3,4が“10”であると,最初のキャラクタ受信時の割り込みが有効となります.またビット3,4が“01”であるとキャラクタ受信時の割り込みが有効となり,すべてのキャラクタの受信時に割り込みが発生します.なお,このときには受信キャラクタにパリティ・エラーが検出されると,特殊受信状態が割り込みとなります.

ビット3,4が“11”のときもキャラクタ受信時の割り込みが有効となりますが,この場合には受信キャラクタにパリティ・エラーが検出されても,特殊受信状態割り込みにはなりません.

■ビット5~7(ウェイト/レディ機能の指定)

これら3ビットによりSIOの$\overline{\mathrm{W/RDYA}}$および$\overline{\mathrm{W/RDYB}}$ピンの使用法を指定します.ビット7が“0”であるとウェイト/レディ機能は用いられず,“1”であるとウェイト機能またはレディ機能のどちらかが用いられます.

ビット6によりウェイト機能,またはレディ機能のうちのどちらかを選択します.ビット6が“0”であるとウェイト機能が選択され,“1”であるとレディ機能が選択されます.

ウェイト/レディ機能はトランスミッタか,レシーバのどちらか一方でしか使用できません.ビット5によりこの選択を行います.すなわち,ビット5が“0”であるとトランスミッタに対してウェイト/レディ機能が適用され,“1”であるとレシーバに対して適用されます.

ウェイト機能を用いると$\overline{\mathrm{W/RDYA}}$,または$\overline{\mathrm{W/RDYB}}$ピンの信号はCPUに対する$\overline{\mathrm{WAIT}}$信号となり,出力はオープン・ドレイン型になります.

この場合,ビット5が“0”の場合には(トランスミッタ),SIOのデータ・レジスタに対してI/Oライト命令(OUT)が出されたときに,バッファ・エンプティでないときには,$\overline{\mathrm{W/RDY(A/B)}}$ピンは“L”になり($\overline{\mathrm{WAIT}}$=“L”)となり,バッファ・エンプティのときにはフローティング状態($\overline{\mathrm{WAIT}}$=“H”)となります.

また,ビット5が“1”の場合には(レシーバ),SIOのデータ・レジスタに対してI/Oリード命令(IN)が出されたときに,レシーバに受信データが用意されていないときには,$\overline{\mathrm{W/RDY(A/B)}}$ピンは“L”になり($\overline{\mathrm{WAIT}}$=“L”),データの用意ができているときにはフローティング状態($\overline{\mathrm{WAIT}}$=“H”)となります.

レディ機能を用いると,$\overline{\mathrm{W/RDYA}}$または$\overline{\mathrm{W/RDYB}}$ピンの信号は,Z80 DMAに対するレディ出力となり,出力はTTLコンパチブルになります.この場合ビット5が“0”の場合には(トランスミッタ),SIOのトランスミッタがバッファ・エンプティのときに,$\overline{\mathrm{W/RDY(A/B)}}$ピンは“L”となり,バッファ・エンプティでないときには“H”となります.

またビット5が“1”の場合には(レシーバ),SIOのレシーバにデータの用意ができているときには,$\overline{\mathrm{W/RDY(A/B)}}$ピンは“L”となり,そうでない場合には“H”となります.したがってレディ機能の場合にSIOが出力するレディ信号の有効レベルは“L”ですので,DMAをそのようにプログラミングする必要があります.

図29にウェイト機能,およびレディ機能を用いた場合の$\overline{\mathrm{W/RDYA}}$および$\overline{\mathrm{W/RDYB}}$ピンの状態を示します.

ライト・レジスタ2(WR2)

図30にライト・レジスタ2(WR2)の構成を示します.このWR2は割り込みベクトルを保持します.SIOはCPUの割り込み認識シーケンス時に,このレジスタの内容を割り込みベクトルとしてCPUに送ります.

しかしながら,ステータス・アフェクツ・ベクトル(WR1のビット2)とプログラミングされているときには,SIOは割り込み要因に応じて,このWR2のビット1~3を変化させます.

したがって,この場合にはWR2にプログラミングした割り込みベクトルのビット1~3は無効であり,その代わりにSIOが発生するビット1~3が用いられます.

このWR2はチャネルBにしかありません.したが

WR1					
D_7	D_6	D_5	$\overline{W}/RDYA$, $\overline{W}/RDYB$ の機能	H/L	状　　態
1	0	0	$\overline{WAIT}$(Tx)	L	トランスミッタがバッファ・エンプティでない
				フローティング	トランスミッタがバッファ・エンプティ
1	0	1	$\overline{WAIT}$(Rx)	L	レシーバにデータの用意ができていない
				フローティング	レシーバにデータの用意ができている
1	1	0	RDY(Tx)	L	トランスミッタがバッファ・エンプティ
				H	トランスミッタがバッファ・エンプティでない
1	1	1	RDY(Rx)	L	レシーバにデータの用意ができている
				H	レシーバにデータの用意ができていない

〈図29〉
ウェイト／レディ機能を用いる場合（WR1, D_7="1"）の $\overline{W}/RDYA$ および $\overline{W}/RDYB$ の状態

って，SIOのチャネルAだけしか使用しない場合でも，割り込みを用いるときには，チャネルBのWR2に対してプログラミングを行う必要があります．

またステータス・アフェクツ・ベクトル機能を用いる場合には，チャネルBのWR1(ビット2）に対しても，プログラミングを行う必要があります．

ライト・レジスタ3(WR3)

図31にライト・レジスタ3（WR3）の構成を示します．このWR3にはレシーバの制御情報をセットします．

■ビット0（イネーブル・レシーバ）

このビットが"1"であるとレシーバがイネーブルされ，レシーバの動作が可能となります．レシーバに関するプログラミングを全部し終わった後に，このビットを"1"にして，レシーバをイネーブルします．

■ビット1(シンク・キャラクタのロードを禁止する)

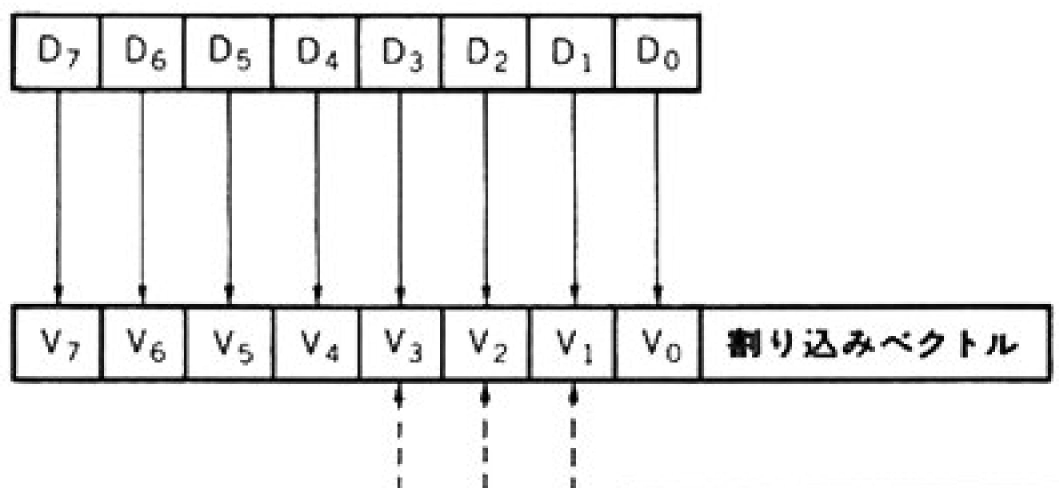
同期モードにおいて，このビットが"1"であると，シンク・キャラクタ（WR6およびWR7のデータ・パターン）が受信されても，レシーバ・バッファ（受信データFIFO）にはロードされません．

このビットはリーディング・シンク（メッセージの先頭のシンク・キャラクタ），およびアイドル・シンク（メッセージの中にあるシンク・キャラクタ）を受信キャラクタから除外するのに用います．

■ビット2(アドレス・サーチ・モード)

このビットはSDLCモードにおいてのみ用いられます．このビットが"1"であると，SIOはアドレス・サーチ・モードに入り，WR6にセットされたアドレス・フィールドを持つメッセージ，またはグローバル・アドレス（FF_{16}）を持つメッセージ以外のメッセージを無視します．

〈図30〉 ライト・レジスタ2（WR2）の構成

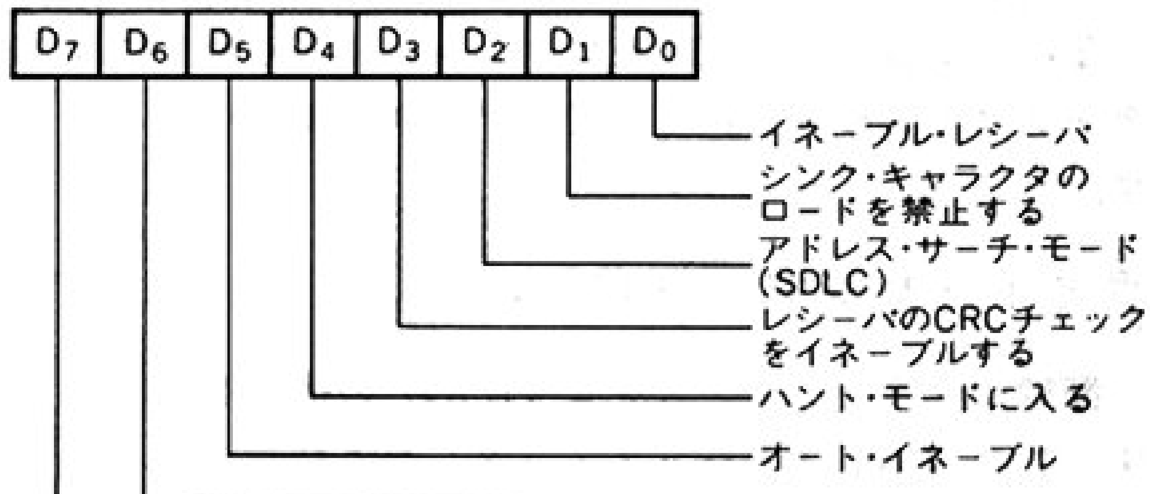


	V3	V2	V1	チャネル	割り込み要因
ステータス・アフェクツ・ベクトル時のベクトル変更	0	0	0	B	トランスミッタ・バッファ・エンプティ
	0	0	1	B	外部/ステータス割り込み
	0	1	0	B	レシーバ・キャラクタ・アベイラブル
	0	1	1	B	特殊受信状態
	1	0	0	A	トランスミッタ・バッファ・エンプティ
	1	0	1	A	外部/ステータス割り込み
	1	1	0	A	レシーバ・キャラクタ・アベイラブル
	1	1	1	A	特殊受信状態

■ビット3（レシーバCRCイネーブル）

このビットが"1"であると，受信シフトレジスタから受信データFIFOに転送された最後のデータからCRC計算が開始されます．

〈図31〉 ライト・レジスタ3（WR3）の構成



D_7	D_6	受信キャラクタ長
0	0	5ビット
0	1	7ビット
1	0	6ビット
1	1	8ビット

■ビット4（エンター・ハント・フェーズ）

このビットが“1”であると，SIOはハント・フェーズ（Hunt Phase）に入ります．このハント・フェーズというのは，同期確立のためにレシーバがシンク・キャラクタ(同期モード)，またはフラグ（SDLCモード）をサーチしている時期のことです．

リセットすると，SIOは自動的にこのハント・フェーズに入ります．しかしながら，何らかの理由により同期状態を失ったとき（同期モード）またはSDLCモードにおいて，受信中のデータが不要になったときなどに，このビットをセットすることにより，同期の再確立をはかったり(同期モード)，次のメッセージの受信に備えたりする（SDLCモード）ことができます．

なお，同期が確立したり(同期モード)，フラグを検出すると，このハント・フェーズは自動的に解除されます．

■ビット5（オート・イネーブル）

このビットを“1”にすると，オート・イネーブル機能が選択され，$\overline{\mathrm{CTS}}$ 入力＝“L”によりトランスミッタがイネーブルされ，$\overline{\mathrm{DCD}}$ 入力＝“L”により，レシーバがイネーブルされます．

この場合，トランスミッタ・イネーブル・コマンド（WR5のビット3），およびレシーバ・イネーブル・コマンド（WR3のビット0）は不要か？　というとそうではなく，これらのコマンドを出しておいても，トランスミッタ・イネーブル・コマンド，およびレシーバ・イネーブル・コマンドは必要です．

なお，オート・イネーブル状態では，トランスミッタ，またはレシーバがイネーブルされているときに，$\overline{\mathrm{CTS}}$ または $\overline{\mathrm{DCD}}$ 入力により，外部からトランスミッタ，またはレシーバのイネーブル状態を制御することができます．

■ビット6，7（受信キャラクタ長）

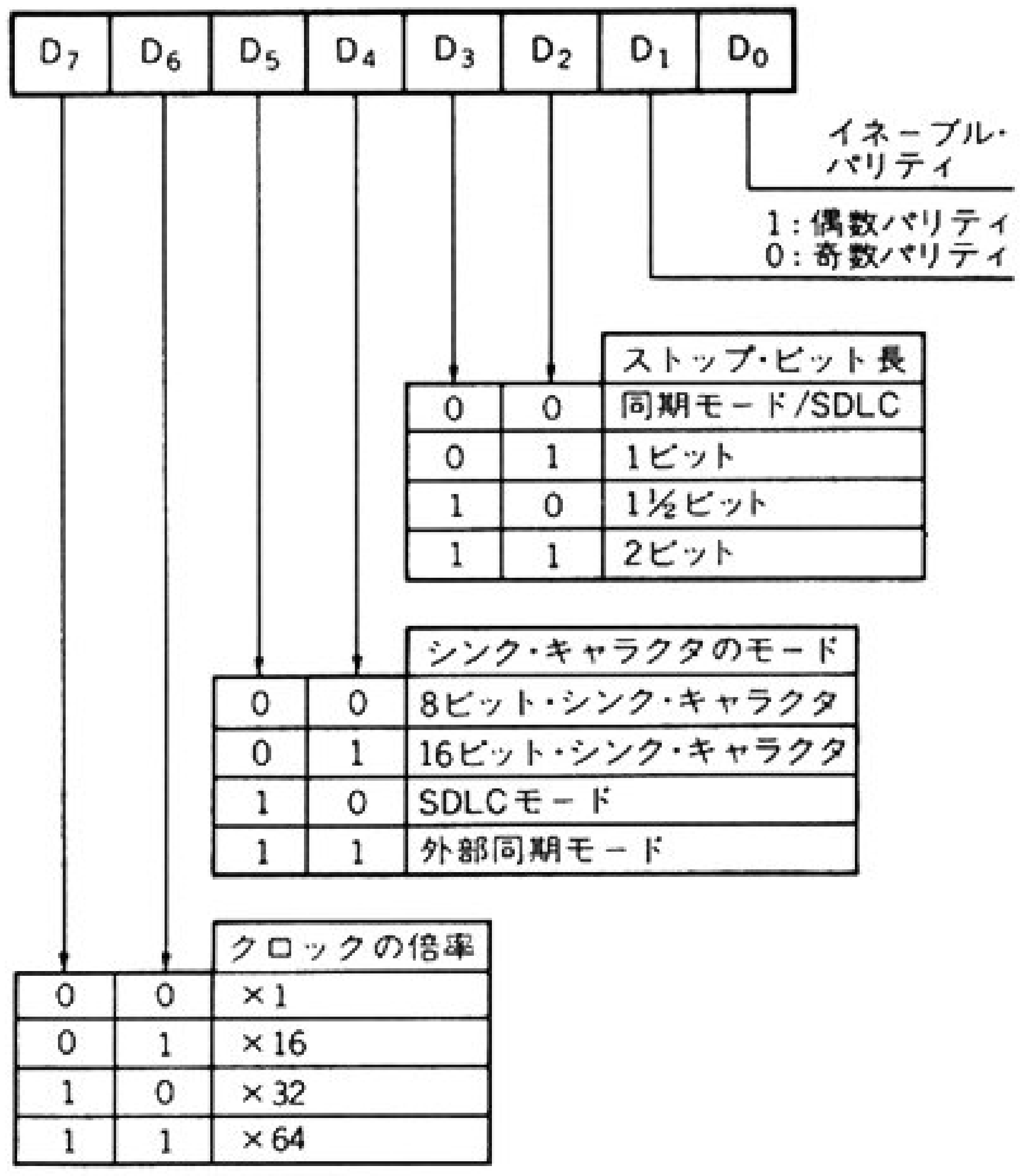
これら2ビットにより受信キャラクタ長を指定します．指定できるのは5～8ビット／キャラクタです．

なお，受信キャラクタ長が7ビット以下の場合には，データは右詰めにされて，受信データFIFOに入ります．

ライト・レジスタ4（WR4）

図32にライト・レジスタ4（WR4）の構成を示します．このWR4はトランスミッタ，およびレシーバ双方に共通なことがらをプログラミングするのに用いられます．

〈図32〉 ライト・レジスタ4（WR4）の構成



■ビット0（イネーブル・パリティ）

このビットが“1”であると，トランスミッタのパリティ・ジェネレータ，およびレシーバのパリティ・チェッカがイネーブルされます．送信時にはトランスミッタはデータの最後にパリティ・ビットを付加します．

受信時にはレシーバは受信キャラクタ長（WR3のビット6，7）で指定されたデータを受信した次のビットをパリティ・ビットとして取り扱います．受信キャラクタ長が8ビットである場合には，パリティ・ビットは受信データFIFOには転送されませんが(できない)，7ビット以下の場合にはデータ・ビットのMSB側にセットされて，受信データFIFOに転送されます．

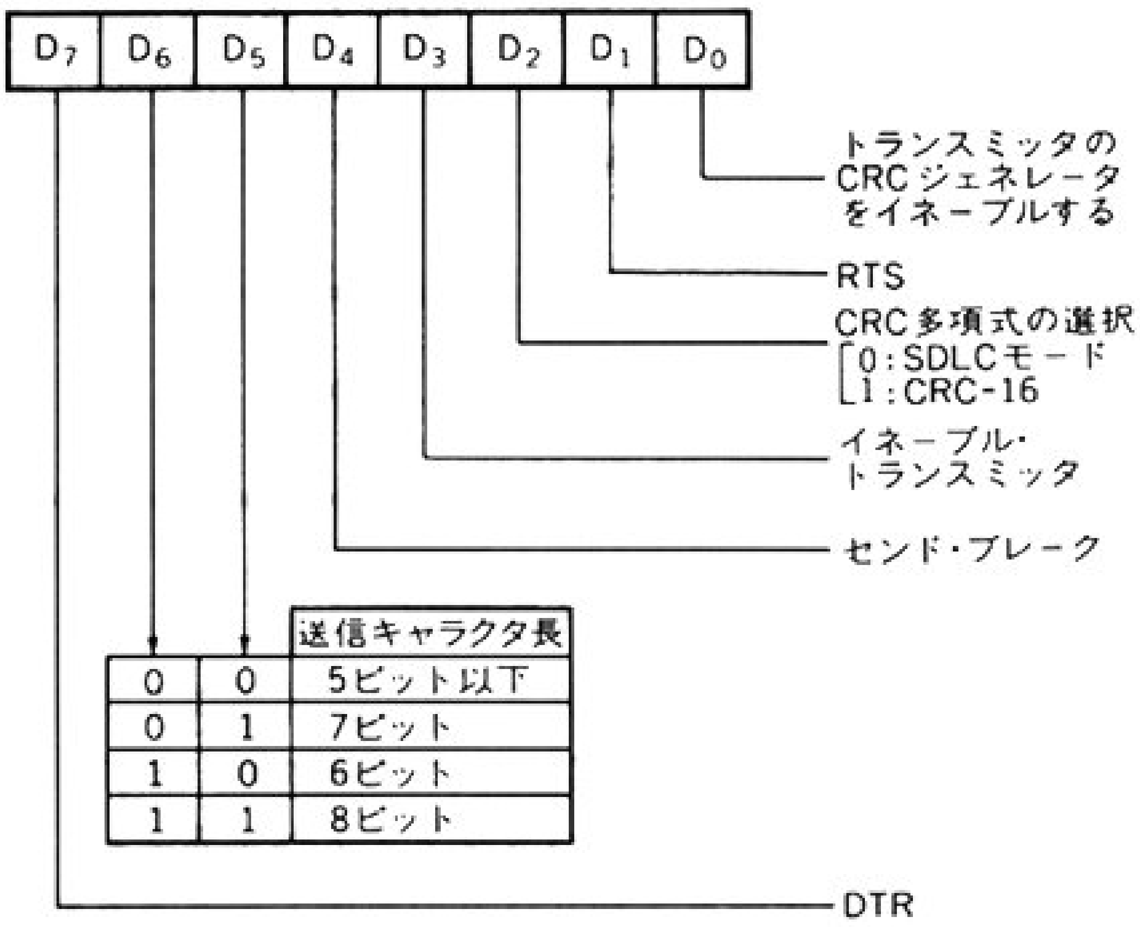
■ビット1（パリティの極性）

このビットが“1”であるとパリティの極性は偶数（Even）となり，“0”であると奇数（Odd）となります．ビット0が“0”であるときには，このビットは無効です．

■ビット2，3（ストップ・ビット長）

これら2ビットにより，非同期モードにおけるストップ・ビット長を指定します．ただしこれら2ビットはモードの選択にも用いられ，ともに“0”であると同期モードまたはSDLCモードとなります．

〈図33〉 ライト・レジスタ5（WR5）の構成



■ビット4，5（シンク・モード）

これら2ビットにより，同期モードにおけるシンク・キャラクタの数（シングル／ダブル）の指定，外部同期モードの指定ならびにSDLCモードの指定を行います．

■ビット6，7（クロックの倍率）

これら2ビットにより，送信クロックおよび受信クロックの倍率を設定します．4種類のクロック倍率を指定できますが，送信クロックおよび受信クロックの倍率は同じでなければなりません．

なお同期モードまたはSDLCモードの場合には，×1のクロック倍率を用いる必要があります．

ライト・レジスタ5（WR5）

図33にライト・レジスタ5（WR5）の構成を示します．このWR5は主としてトランスミッタの制御に用いられますが，ビット2だけは例外で，レシーバの制御にも適用されます．

■ビット0（トランスミッタCRCイネーブル）

このビットが"1"であると，トランスミッタのCRCジェネレータがイネーブルされます．送信データが送信データ・バッファから送信シフトレジスタにロードされたときにこのビットが"1"であれば，そのデータに対してCRC計算が行われます．

またこのビットが"1"であると，同期モードまたはSDLCモードにおいて，トランスミッタ・アンダラン状態になると，CRCが自動的に送信されます．

■ビット1（RTS：リクエスト・ツー・センド）

このビットを"1"にすると，SIOの$\overline{RTS}$ピンの状態は"L"になります．またこのビットを"0"にするとRTSピンの状態は"H"になります．

非同期モードではRTSピンの状態が実際に"H"になるのは，送信バッファが空になったとき（トランスミッタ・バッファ・エンプティ）ですが，同期モードおよびSDLCモードでは送信バッファの状態に関係なく，このビットの状態にそのまましたがいます．

■ビット2（CRC-16/$\overline{SDLC}$）

このビットによりCRCチェッカ／ジェネレータの多項式の選択を行います．このビットが"1"であるとCRC-16（同期モード）が選択され，"0"であるとSDLC多項式が選択されます．

■ビット3（イネーブル・トランスミッタ）

このビットが"1"であると，トランスミッタがイネーブルされます．

■ビット4（センド・ブレーク）

このビットが"1"であると，送信データ・ライン（TxDA/TxDB）の状態が強制的にスペース状態("L")になります．

■ビット5，6（送信キャラクタ長）

これら2ビットにより送信データのキャラクタ長を指定します．指定できるキャラクタ長は6～8ビット／キャラクタの3種類と，5ビット／キャラクタです．SIOに対して送られる送信データは，右詰めになっている必要があります．

キャラクタ長が6～8ビットのときには，ビット5，および6によりSIOはキャラクタ長を知ることができますので，SIOに対して送られる送信データ中の不要のビットの値はどうでもよいのですが，キャラクタ長が5ビット以下の場合には，SIOは5ビット以下であることはわかるけれども，何ビットをデータとして解釈すればよいのかわかりません．

この問題を解決するためにSIOでは，キャラクタ長が5ビット以下のときにはデータの中にキャラクタ長に関する情報を含めています．図34にキャラクタ長が5ビット以下の場合のデータ・フォーマットを示します．

この図からわかるように，ビット4～7によりキャラクタ長（1～5ビット）がわかるように，フォーマッティングされています．

■ビット7（DTR：データ・ターミナル・レディ）

このビットを"1"にすると，SIOの$\overline{DTR}$ピンの状態が"L"になり，"0"にするとDTRピンの状態

〈図34〉 キャラクタ長が5ビット以下の場合の送信データ・フォーマット

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	キャラクタ長（ビット）
1	1	1	1	0	0	0	D	1
1	1	1	0	0	0	D	D	2
1	1	0	0	0	D	D	D	3
1	0	0	0	D	D	D	D	4
0	0	0	D	D	D	D	D	5

は“H”になります.

ライト・レジスタ6(WR6)

図35にライト・レジスタ（WR6）の構成を示します.このレジスタには次のような情報がセットされます.

① 送信シンク・キャラクタ（シングル・シンクおよび外部同期の場合）

② 第1のシンク・キャラクタ（ダブル・シンクの場合）

③ 従局アドレス（SDLCモードの場合）

ライト・レジスタ7(WR7)

図36にライト・レジスタ7（WR7）の構成を示します.このレジスタには次のような情報がセットされます.

① 受信シンク・キャラクタ(シングル・シンクの場合)

② 第2のシンク・キャラクタ(ダブル・シンクの場合)

③ フラグ・パターン：$7E_{16}$（SDLCモードの場合）

なお，外部同期のときにはこのレジスタは用いられません.図37にWR6およびWR7の使われかたをまとめて示します.

リード・レジスタ

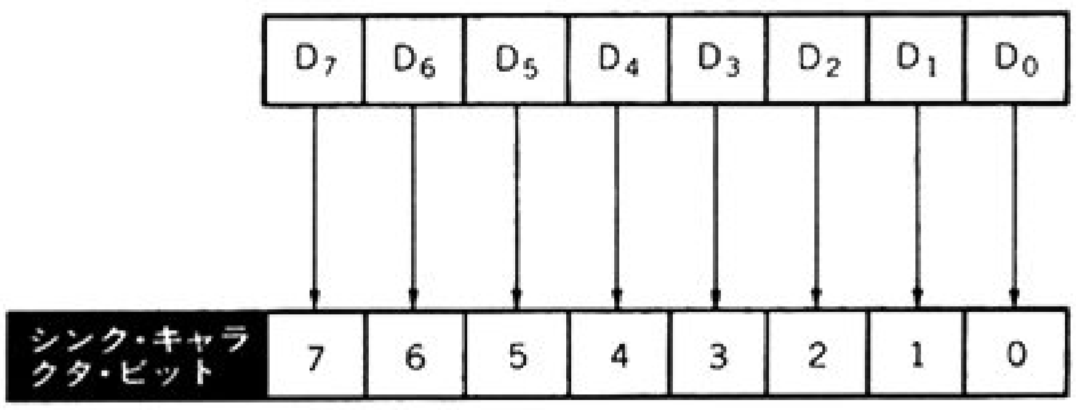
リード・レジスタ0(RR0)

図38にリード・レジスタ0（RR0）の構成を示します.このRR0には送受信バッファの状態（送信バッファ・エンプティ，受信バッファ・アベイラブル）および外部／ステータス割り込みの原因となりうる状態がセットされます.

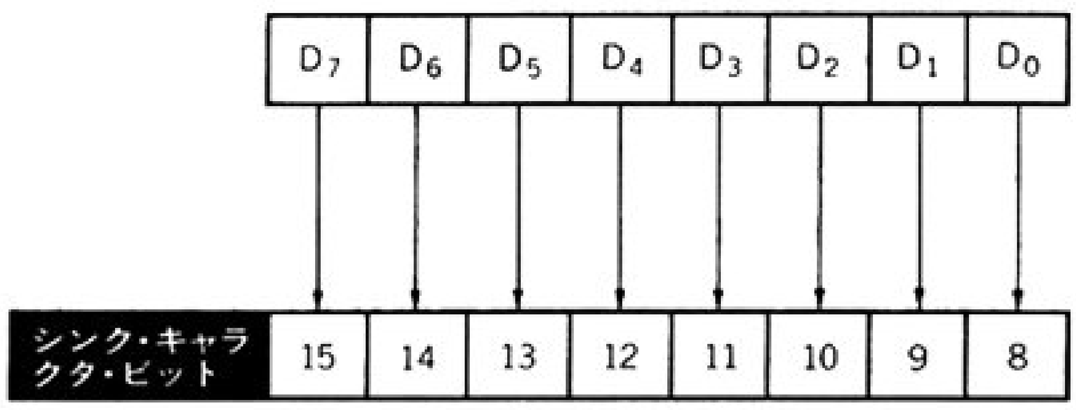
ステータス・アフェクツ・ベクトルと指定しないで割り込みを起こさせる場合には，割り込みルーチンの入口にてこのレジスタの内容を読み取り，割り込み要因をチェックする必要があります.また外部／ステータス割り込みが発生したときには，このレジスタを読んでその原因を知る必要があります.

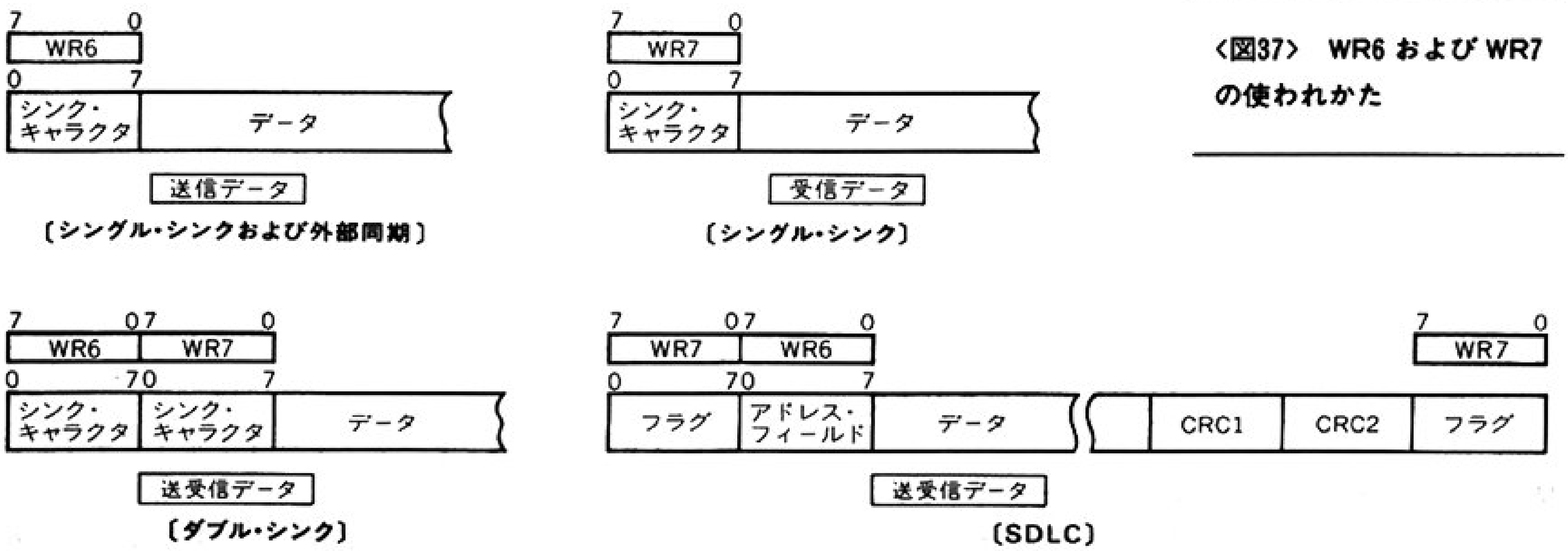
さらに割り込みを使用せず，ポールド・モードでSIOをコントロールする場合には，このレジスタの内容によりSIOの状態を知り，必要な処理を行う必要があります.

〈図35〉 ライト・レジスタ6（WR6）の構成



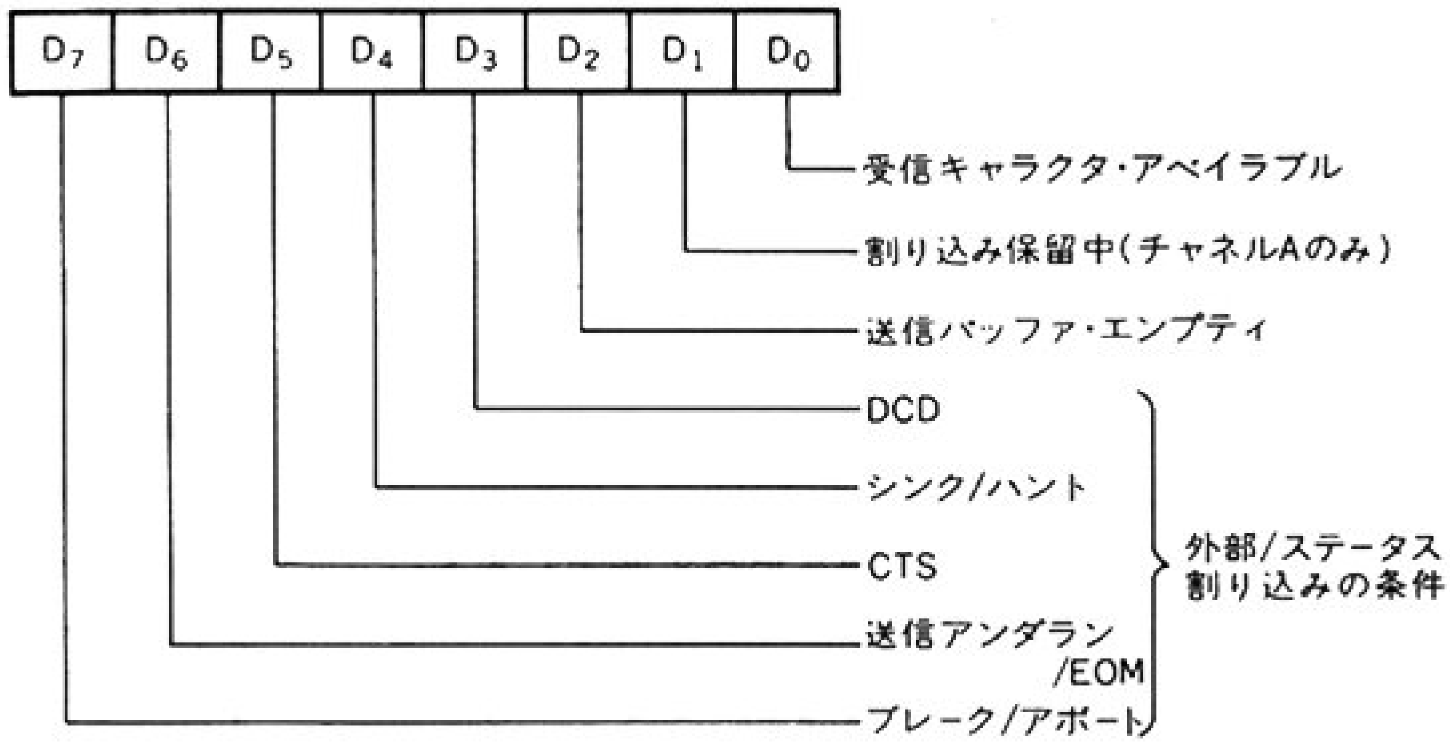
〈図36〉 ライト・レジスタ7（WR7）の構成





〈図37〉 WR6およびWR7の使われかた

〈図38〉 リード・レジスタ0（RR0）の構成



■ビット0（受信キャラクタ・アベイラブル）

このビットは受信データFIFOに1バイト以上のデータが用意されている場合にセット（"1"）され，同FIFOが完全に「空」であるとリセット（"0"）されます．

■ビット1（割り込み保留中）

このビットはSIOが割り込みを発生すべき何らかの状態を持っているときにセットされ，そうでない場合にはリセットされます．このビットはチャネルAのRR0においてのみ有効です．

■ビット2（トランスミッタ・バッファ・エンプティ）

このビットは送信データ・バッファが「空」であるとセットされます．ただし同期モードおよびSDLCモードにおいて，CRCが送信されるときにはセットされません．送信データ・バッファにデータをロードするとこのビットはリセットされます．SIOはリセットされると，このビットをセットします．

■ビット3（DCD：データ・キャリヤ・デテクト）

このビットはSIOの $\overline{\mathrm{DCDA}}$/$\overline{\mathrm{DCDB}}$ ピンの状態を示しています．$\overline{\mathrm{DCDA}}$/$\overline{\mathrm{DCDB}}$ ピンの状態が"L"であれば"1"となり，"H"であれば"0"となります．

これらのピンの状態は常に読み込み可ですが，外部／ステータス割り込みが発生すると，これらのピンの状態はラッチされ，以後状態が変化してもこのビットの値は変わりません．

$\overline{\mathrm{DCDA}}$/$\overline{\mathrm{DCDB}}$ の状態変化により外部／ステータス・割り込みが発生したときには，その変化がL → Hであるときにはこのビットは"0"となり，H → Lのときには"1"となります．

前述のラッチ状態を解除し，その時点の $\overline{\mathrm{DCDA}}$/$\overline{\mathrm{DCDB}}$ ピンの状態を知りたいときには，リセット外部／ステータス割り込みコマンド（WR0：10_{16}）を出した後に，このRR0を読み込みます．この項の説明はDCDだけに限らず，他の外部／ステータス割り込みの要因となる状態についても適用されます．

■ビット4（シンク／ハント：Sync/Hunt）

このビットの意味はSIOがプログラミングされている通信モードに応じて異なります．したがって通信モードごとにこのビットの意味を説明します．

●非同期モード：非同期モードではこのビットは単にSIOの $\overline{\mathrm{SYNC}}$ ピンの状態または状態変化を示します．

●外部同期モード：外部同期モードではSIOの $\overline{\mathrm{SYNC}}$ ピンは入力となり，このピンには外部同期検出ロジックからの同期確立信号が接続されます．したがってこのビットには同信号の状態または状態変化がセットされます．

●内部同期モード：内部同期モードでは，エンター・ハント・フェーズ・コマンド（WR3のビット4）によりこのビットはセット（"1"）され，SIOがシンク・キャラクタを検出して同期確立ができたときに，リセット（"0"）されます．

●SDLCモード：SDLCモードの場合にも，エンター・ハント・フェーズ・コマンドにより，このビットはセットされます．そしてオープニング・フラグを検出すると，このビットはリセットされます．

■ビット5（CTS：クリヤ・ツー・センド）

このビットの意味は，DCDをCTSに読み変えればビット3の説明がそのまま適用できます．

■ビット6（トランスミッタ・アンダラン／EOM）

SIOがリセットされると，このビットはセット状態（"1"）になります．そしてリセット・トランスミッタ・アンダラン/EOMラッチ・コマンド（WR0，D_6＝1，D_7＝1）を出すと，リセットされます．さらにトランスミッタ・アンダラン状態が発生すると，このビットはセットされ，同時に外部／ステータス割り込みが発生します．

このビットは同期モードおよびSDLCモードにおける送信動作の終結，つまりCRCの送信を行うのに用いられます．同期モードではこのビットがセットされているときに，トランスミッタ・アンダラン状態が発生しても外部／ステータス割り込みを起こさず，代わりにアイドル・シンク・キャラクタを送信してデー

タ・ラインの同期を維持します.

そしてトランスミッタ・バッファ・エンプティの割り込みが発生したときに, 最後のデータ, たとえばETXがすでに送信されていたような場合には, プログラムは前述のコマンドによりこのビットをリセットします.

このビットがリセットされていると, SIOは送信バッファにデータがなければCRC(2キャラクタ)を送信します. CRCを送り終わるとSIOはこのビットをセットし, 外部/ステータス割り込みを発生します.

このときプログラムがSIOに対して何もデータをロードしなければ, SIOはアイドル・シンク・キャラクタを出し続けます. プログラムがデータ, たとえばパッド・キャラクタをロードすれば, SIOはそのキャラクタを送信し, これを送信し終わるとバッファ・エンプティの割り込みを発生します.

SDLCモードではこのビットがセットされているときに, トランスミッタ・バッファ・エンプティの状態であると, SIOはフラグを送信し続けます. しかしながら, SDLCモードでは同期モードの場合のようなアイドル・キャラクタという概念がありませんので, 最初のデータをロードしたときに, このビットをリセットしておきます.

このビットがリセットされていると, SIOはトランスミッタ・アンダーラン状態になったときにCRCを送信し, 同時にこのビットをセットして外部/ステータス割り込みを発生します. CRCが完全に送信し終わると, SIOはバッファ・エンプティの割り込みを発生します. ここで送信割り込みをリセットするコマンド(WR0, コマンド5)を出し, SIOに対して何もデータをロードしなければクロージング・フラグが送信されます.

■ビット7 (ブレーク/アボート)

このビットは非同期モードでは, ブレーク状態検出の意味を持ち, SDLCモードでは, アボート・シーケンスの意味を持ちます.

SIOはブレーク状態を検出すると, このビットをセットし, 外部/ステータス割り込みを発生します. 割り込みルーチンが, リセット外部/ステータス・コマンドを出すことにより, ブレーク検出状態をクリヤしておくと, ブレーク状態が解除されたときにも, 外部/ステータス割り込みを発生します.

SDLCモードにおいては, アボート・シーケンスを検出すると, このビットをセットして外部/ステータス割り込みを発生します. 割り込みルーチンがリセット外部/ステータス割り込みコマンドを出しておけば, アボート状態が解除されたときも, 外部/ステータス割り込みが発生します.

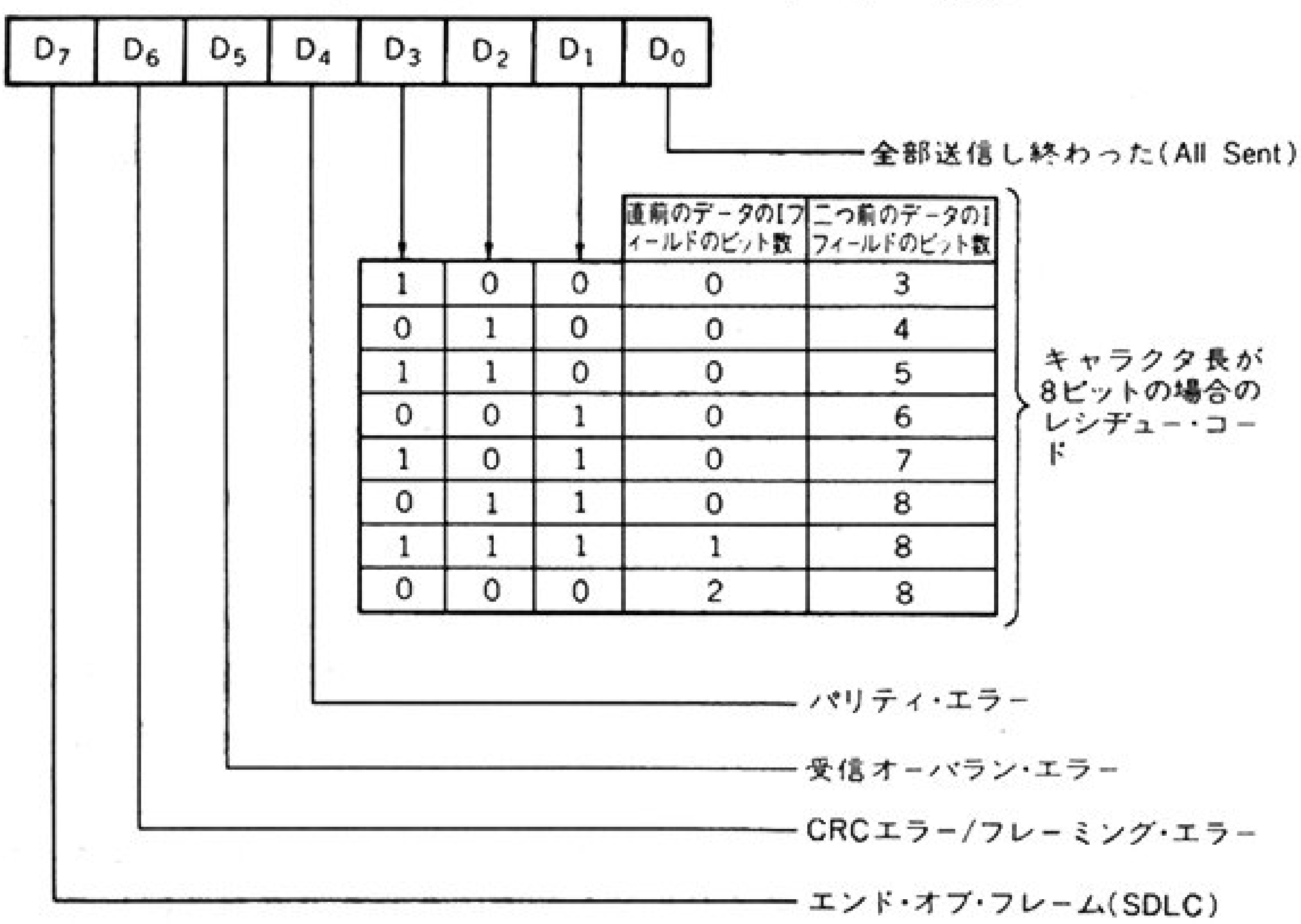
リード・レジスタ1(RR1)

図39にリード・レジスタ1 (RR1) の構成を示します. このレジスタには, 特殊受信状態割り込みの原因と, SDLCモードにおけるIフィールドのレシデュー(Residue:半端(はんぱ)なビット)・コードがセットされます.

■ビット0 (全部送信し終わった:All Sent)

非同期モードにおいて, 全キャラクタの送信が終わり, SIO中に送信データがまったくなくなったときにこのビットはセットされます. 同期モードおよびSDLCモードでは, このビットは常にセットされています.

〈図39〉 リード・レジスタ1 (RR1) の構成



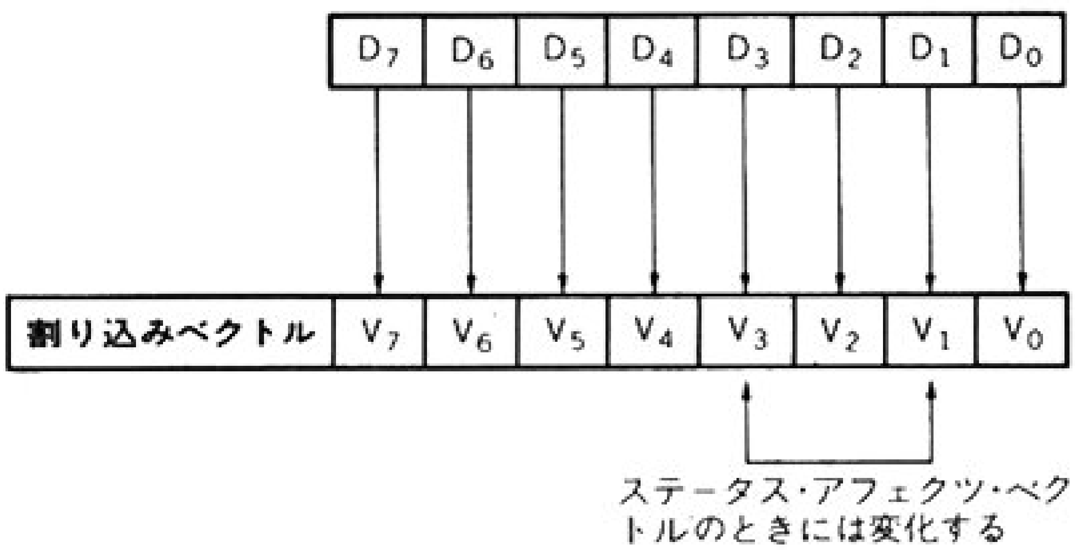
D_3	D_2	D_1	直前のデータのIフィールドのビット数	二つ前のデータのIフィールドのビット数
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

■ビット1〜3 (レシデュー・コード0〜2)

これらのビットはSDLCモードにおけるIフィールドの最後の半端なビット数を示す3ビットのコードであり, レシデュー・コード(Residue Code)と呼びます. 通常の場合, 受信するビット数の総和はキャラクタ長の整数倍でありますが, 整数倍でなかった場合, このレシデュー・コードにより最後の半端なビット数を示します.

このレシデュー・コードが有効であるのは, エンド・オブ・フレーム状態(RR1のビット7)にあるときだ

〈図40〉 リード・レジスタ2（RR2）の構成



けです．図39にはキャラクタ長が8ビットである場合のレシデュー・コードが示されています．

■ビット4（パリティ・エラー）

このビットはパリティがイネーブルされているときに（WR4のビット0），受信データ中にパリティ・エラーが検出されるとセットされます．このビットはSIO内部でラッチされ，いったんエラーが検出されると，リセット・エラー・コマンド（WR0のコマンド6）が出されるまで，その状態を保ちます．

したがってメッセージを全部受信し終わってから，その中にパリティ・エラーが検出されたデータがあったか否かをチェックすることができます．

■ビット5（オーバラン・エラー）

このビットは3キャラクタのデータが受信データFIFOに存在するときに，さらに次のデータが受信されるとエラーFIFOにセットされます．このビットが実際にRR1中にセットされるのは，エラーFIFOにこのビットがセットされているデータが読み取られたときです．

このエラー状態はラッチされますので，リセット・エラー・コマンドが出されるまでリセットされません．

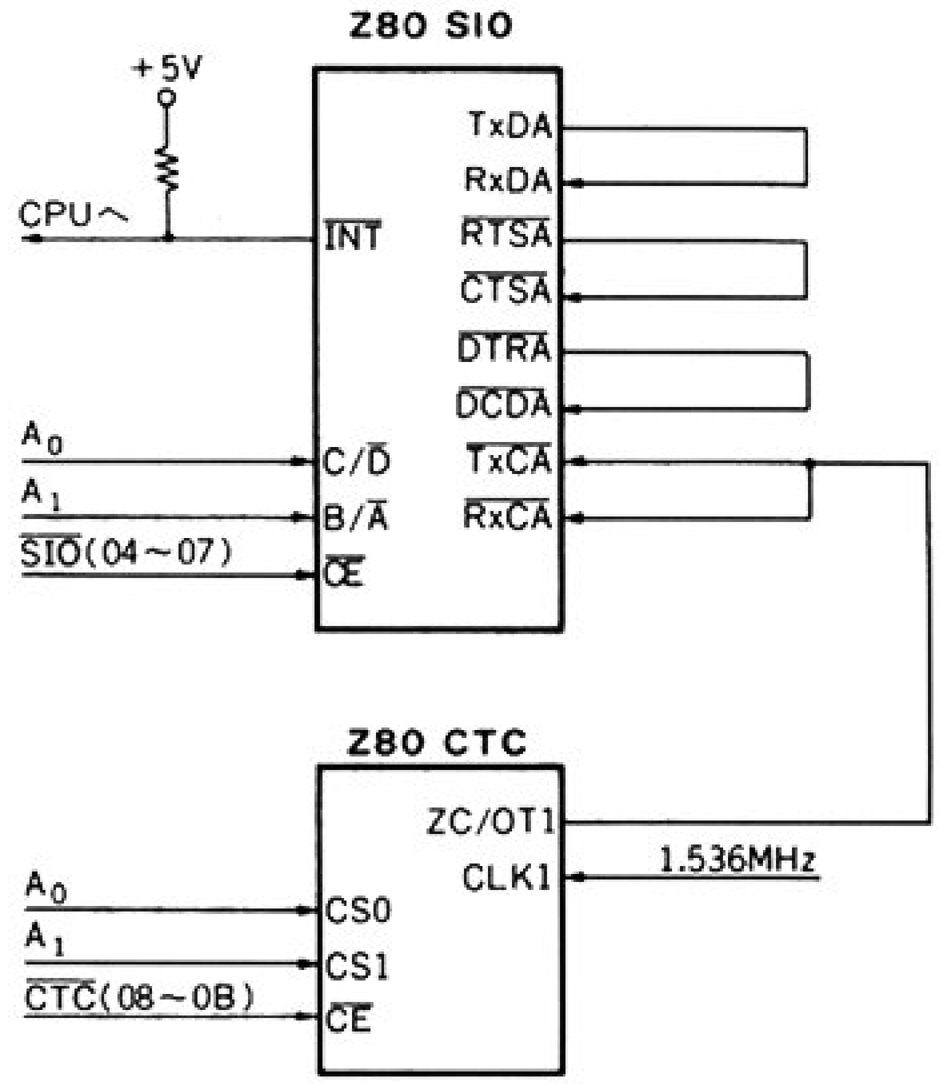
■ビット6（CRC／フレーミング・エラー）

非同期モードでは，受信データにフレーミング・エラーが検出されると，このビットがセットされます．同期モードおよびSDLCモードでは，このビットはそれまでに受信されたデータに対するCRCチェック結果が示されています．これらの状態はラッチされず，キャラクタが受信されるごとに変化します．

■ビット7（エンド・オブ・フレーム）

このビットはSDLCモードにおいてのみ用いられ，受信データ中にクロージング・フラグが検出されたことを示します．このビットはリセット・エラー・コマンドにより，リセットされますが，次のフレームの最初のキャラクタを受信することにより，リセットされ

〈図42〉 SIOのループ・バック構成



ます．

リード・レジスタ2（RR2）

図40にリード・レジスタ2（RR2）の構成を示します．このレジスタに対する読み取りはチャネルBにのみ有効です．ステータス・アフェクツ・ベクトルでないときには，このレジスタの内容はチャネルBのWR2の内容と一致します．

ステータス・アフェクツ・ベクトルのときには，その時点に保留中の割り込みのうち，最高の優先順位を持つ割り込みベクトルがセットされます．保留割り込みがない場合には，V_3="0"，V_2="1"，V_1="1"となります．

＊

以上でZ80 SIOのコントロール・コマンドの説明を終わりますが，図41にWR0～7，およびRR0～2の各レジスタをまとめておきます．

Z80 SIOのプログラミング

ここではSIOの送信データ・ライン（TxDA）と受信データ・ライン（RxDA），$\overline{RTSA}$と$\overline{CTSA}$，$\overline{DTRA}$と$\overline{DCDA}$を互いに接続したループ・バック構成により，SIOを実際に動作させるプログラム例をいくつか示します．

図42に本項で示すプログラム（Prog❶～❺，❼）が動作するための構成を示します．なお，Prog❻が動作するための構成については，同プログラムの説明の

〈図41〉 Z80 SIOのコマンド一覧

●WR0●

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

ポインタ（D_2 D_1 D_0）

D_2	D_1	D_0	ポインタ
0	0	0	レジスタ0
0	0	1	レジスタ1
0	1	0	レジスタ2
0	1	1	レジスタ3
1	0	0	レジスタ4
1	0	1	レジスタ5
1	1	0	レジスタ6
1	1	1	レジスタ7

コマンド（D_5 D_4 D_3）

D_5	D_4	D_3	コマンド番号	コマンド
0	0	0	0	ノー・オペレーション
0	0	1	1	センド・アボート(SDLC)
0	1	0	2	外部/ステータス割り込みのリセット
0	1	1	3	チャネル・リセット
1	0	0	4	次のキャラクタ受信時の割り込みを可とする
1	0	1	5	保留中の送信割り込みをリセットする
1	1	0	6	エラー・リセット
1	1	1	7	RETI(チャネルAのみ)

D_7 D_6

D_7	D_6	
0	0	ノー・オペレーション
0	1	受信CRCチェッカのリセット
1	0	送信CRCジェネレータのリセット
1	1	送信アンダラン/EOMラッチのリセット

●WR2●

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0

割り込みベクトル

ステータス・アフェクツ・ベクトル時のベクトル変更（V_3 V_2 V_1）

V_3	V_2	V_1	チャネル	割り込み要因
0	0	0	B	トランスミッタ・バッファ・エンプティ
0	0	1	B	外部/ステータス割り込み
0	1	0	B	レシーバ・キャラクタ・アベイラブル
0	1	1	B	特殊受信状態
1	0	0	A	トランスミッタ・バッファ・エンプティ
1	0	1	A	外部/ステータス割り込み
1	1	0	A	レシーバ・キャラクタ・アベイラブル
1	1	1	A	特殊受信状態

●WR1●

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

- D_0：外部/ステータス割り込み可("1"のとき)
- D_1：送信割り込み可("1"のとき)
- D_2：ステータス・アフェクツ・ベクトル("1"のとき)(チャネルBのみ)

D_4	D_3	
0	0	受信割り込み不可
0	1	最初のキャラクタ受信時の割り込み
1	0	キャラクタ受信時の割り込み(パリティ・エラーは特殊受信状態割り込みになる)
1	1	キャラクタ受信時の割り込み(パリティ・エラーは特殊受信状態割り込みにならない)

- D_5：0:送信時にウェイト/レディ機能を用いる　1:受信時にウェイト/レディ機能を用いる
- D_6：0:ウェイト機能　1:レディ機能
- D_7：0:ウェイト/レディ機能を使わない　1:ウェイト/レディ機能を使う

●WR3●

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

- D_0：イネーブル・レシーバ
- D_1：シンク・キャラクタのロードを禁止する
- D_2：アドレス・サーチ・モード(SDLC)
- D_3：レシーバのCRCチェックをイネーブルする
- D_4：ハント・モードに入る
- D_5：オート・イネーブル

D_7	D_6	受信キャラクタ長
0	0	5ビット
0	1	7ビット
1	0	6ビット
1	1	8ビット

●WR4●

- D_0: イネーブル・パリティ
- D_1: 1:偶数パリティ / 0:奇数パリティ

D_3	D_2	ストップ・ビット長
0	0	同期モード/SDLC
0	1	1ビット
1	0	1½ビット
1	1	2ビット

D_5	D_4	シンク・キャラクタのモード
0	0	8ビット・シンク・キャラクタ
0	1	16ビット・シンク・キャラクタ
1	0	SDLCモード
1	1	外部同期モード

D_7	D_6	クロックの倍率
0	0	×1
0	1	×16
1	0	×32
1	1	×64

●WR5●

- D_0: トランスミッタのCRCジェネレータをイネーブルする
- D_1: RTS
- D_2: CRC多項式の選択 [0:SDLCモード / 1:CRC-16]
- D_3: イネーブル・トランスミッタ
- D_4: センド・ブレーク
- D_7: DTR

D_6	D_5	送信キャラクタ長
0	0	5ビット以下
0	1	7ビット
1	0	6ビット
1	1	8ビット

●WR6●

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
シンク・キャラクタ・ビット	7	6	5	4	3	2	1	0

●WR7●

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
シンク・キャラクタ・ビット	15	14	13	12	11	10	9	8

●RR0●

- D_0: 受信キャラクタ・アベイラブル
- D_1: 割り込み保留中(チャネルAのみ)
- D_2: 送信バッファ・エンプティ
- D_3: DCD
- D_4: シンク/ハント
- D_5: CTS
- D_6: 送信アンダラン/EOM
- D_7: ブレーク/アボート

(D_3〜D_7: 外部/ステータス割り込みの条件)

●RR1●

- D_0: 全部送信し終わった(All Sent)
- D_3 D_2 D_1: キャラクタ長が8ビットの場合のレシデュー・コード

D_3	D_2	D_1	直前のデータのIフィールドのビット数	二つ前のデータのIフィールドのビット数
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

- D_4: パリティ・エラー
- D_5: 受信オーバラン・エラー
- D_6: CRCエラー/フレーミング・エラー
- D_7: エンド・オブ・フレーム(SDLC)

●RR2●

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
割り込みベクトル	V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0

V_3〜V_1: ステータス・アフェクツ・ベクトルのときには変化する

<Prog ❶> Z80 CTC を用いたボーレート・ジェネレータ

```
                          ;
                          ;       SIMPLE BAUD RATE GENERATOR
                          ;
                          ;  THIS BAUD RATE GENERATOR USES Z80 CTC CHANNEL 1 AND
                          ;  OUTPUTS CLOCK OF WHICH FREQUENCY IS 9600 X 16.
                          ;  INPUT CLOCK FREQUENCY TO CTC CHANNEL 1 IS 1.536 MHZ.
                          ;  CTC CHANNEL 1 IS PROGRAMMED AS COUNTER MODE.
                          ;
0009                      CTC1   EQU   09H          ;CTC CHANNEL 1 ADDRESS
                          ;
                                 ORG   2000H
2000    3E 47                    LD    A,47H        ;INHIBIT INTERRUPT
                                                    ;COUNTER MODE
                                                    ;LOAD TIME CONSTANT
                                                    ;RESET
2002    D3 09                    OUT   (CTC1),A     ;MODE WORD TO CTC CHNL 1
2004    3E 0A                    LD    A,10         ;TIME CONSTANT=10
2006    D3 09                    OUT   (CTC1),A     ;LOAD TIME CONSTANT
                          ;
                          ;      BAUD RATE GENERATOR BEGINNS RUNNING
                          ;
                                 END
```

項にて述べます.

以下に示すプログラムではZ80 CTCのチャネル1をカウンタ・モードとしてプログラムし，その出力をSIOの送信クロック（$\overline{TxCA}$）および受信クロック（$\overline{RxCA}$）として用いています．Prog ❶ にCTCによるボーレート・ジェネレータを示します．

このプログラムはCTCのチャネル1をカウンタ・モードに設定し，その入力クロック（CLK 1 = 1.536 MHz）を10分の1しています．したがって，チャネル1の出力（ZC/TO1）には153.6kHz（9600×16Hz）のパルスが得られます．このパルスはSIOを非同期モードに設定し，16倍のクロックにて9600BPSで動作させるための入力クロックとなります．

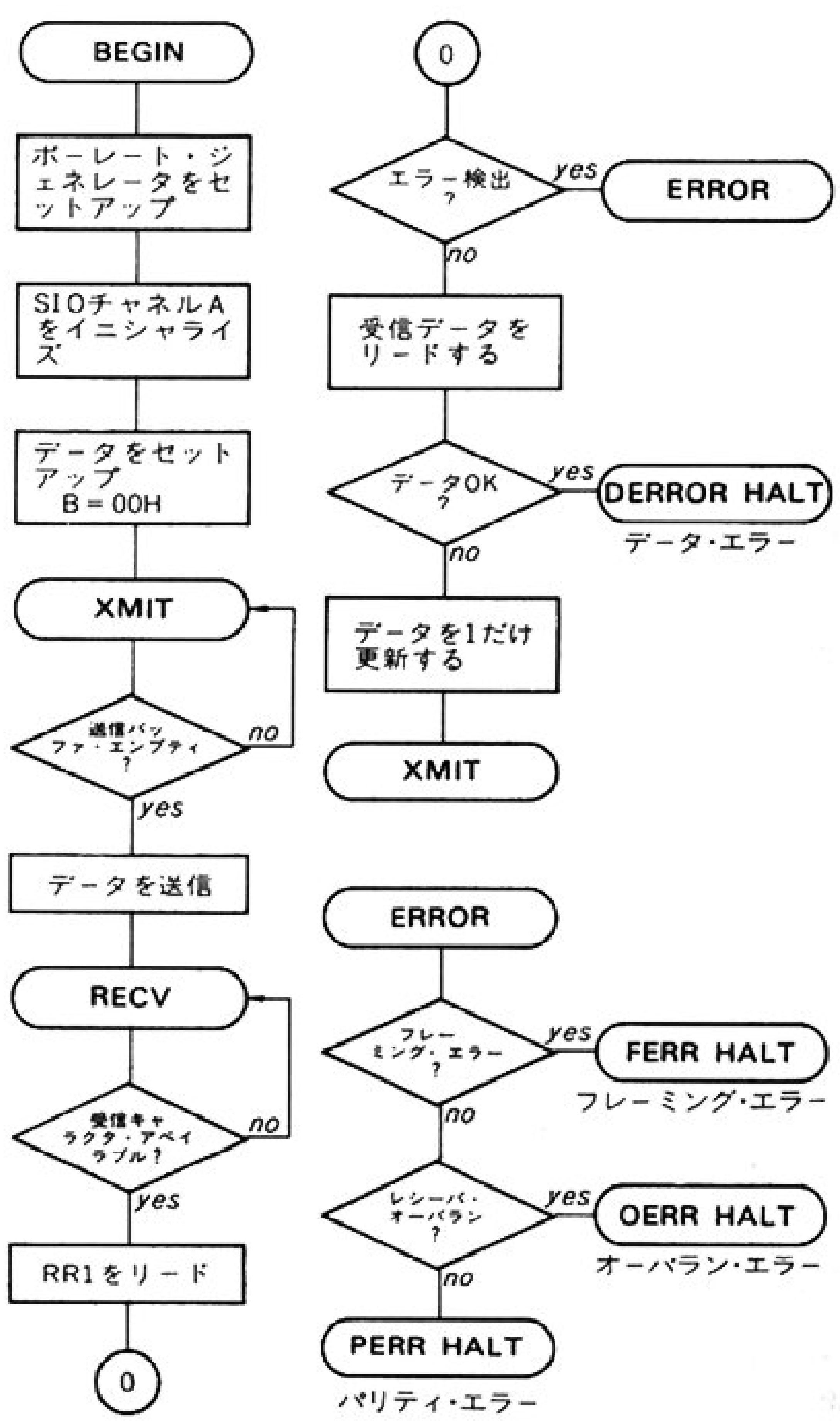
SIOを同期モード，またはSDLCモードで動作させるときには，CTCのタイム・コンスタントを160とすることにより，その出力として9600Hzのパルスが得られ，これをSIOの送・受信クロックとして用います．

SIO ループ・バック・テスト（非同期モード，ポーリング）

SIOを非同期モードに設定し，割り込みを用いず，ポーリングによりデータのループ・バック・テストを行う例を Prog ❷ に示します．また図43にこのプログラムのフロー・チャートを示します．

このプログラムはまずボーレート・ジェネレータ（CTC）をセットアップします．この部分は Prog ❶ とまったく同じです．次にプログラムはSIOをイニシャライズし，データ（レジスタB）をセットアップした後にループ・バック・テストに入ります．

<図43> Prog ❷ のフロー・チャート
（SIO ループ・バック・テスト，非同期モード，ポーリング）



〈Prog❷〉 Z80 SIO ループ・バック・テスト（非同期モード，ポーリング）

```
                        ;       Z80 SIO LOOP BACK TEST (ASYNC,POLLING)
                        ;
                        ;       THIS PROGRAM TRANSMITS DATA TO SIO CHANNEL A
                        ;       AND RECEIVES LOOP-BACKED DATA FROM THE SAME
                        ;       CHANNEL.
                        ;
0009                    CTC1    EQU     09H             ;CTC CHANNEL 1
                        ;
0004                    SIOAD   EQU     04H             ;SIO CHANNEL A DATA
0005                    SIOAC   EQU     SIOAD+01H       ;SIO CHANNEL A CONTROL
0006                    SIOBD   EQU     SIOAC+01H       ;SIO CHANNEL B DATA
0007                    SIOBC   EQU     SIOBD+01H       ;SIO CHANNEL B CONTROL
                        ;
0000                            ASEG
                                ORG     2000H
2000    F3              BEGIN:  DI                      ;DISABLE INTERRUPT
2001    31 208E                 LD      SP,STACK        ;SET STACK POINTER
                        ;       SET UP BAUD RATE GENERATOR
2004    3E 47                   LD      A,47H           ;MODE WORD FOR CTC
2006    D3 09                   OUT     (CTC1),A        ;LOAD MODE WORD TO CTC
2008    3E 0A                   LD      A,10            ;SET TIME CONSTANT
200A    D3 09                   OUT     (CTC1),A        ;LOAD TIME CONSTANT
                        ;       INITIALIZE SIO CHANNEL A
200C    21 2045                 LD      HL,SIOACC       ;LOAD COMMAND ADDRESS
200F    06 09                   LD      B,SIOACL-SIOACC ;LOAD COMMAND LENGTH
2011    0E 05                   LD      C,SIOAC         ;LOAD SIO A CTL ADRS
2013    ED B3                   OTIR                    ;LOAD COMMAND TO SIO
                        ;       SET UP DATA AND SIO A DATA PORT ADDRESS
                        ;       B=DATA TO BE TRANSMITTED
                        ;       C=SIO CHANNEL A DATA ADDRESS
2015    06 00                   LD      B,0             ;SET UP DATA (00H)
2017    0E 04                   LD      C,SIOAD         ;LOAD DATA ADDRESS
                        ;       TRANSMIT DATA
2019    DB 05           XMIT:   IN      A,(SIOAC)       ;READ RR0
201B    CB 57                   BIT     2,A             ;TX BUFFER EMPTY ?
201D    28 FA                   JR      Z,XMIT          ;NO,XMIT
201F    ED 41                   OUT     (C),B           ;TRANSMIT DATA
                        ;       RECEIVE DATA
2021    DB 05           RECV:   IN      A,(SIOAC)       ;READ RR0
2023    CB 47                   BIT     0,A             ;RX CHAR. AVAILABLE ?
2025    28 FA                   JR      Z,RECV          ;NO,RECV
2027    3E 01                   LD      A,1             ;TO ACCESS RR1
2029    D3 05                   OUT     (SIOAC),A       ;
202B    DB 05                   IN      A,(SIOAC)       ;READ RR1
202D    E6 70                   AND     70H             ;ANY ERROR DETECTED ?
202F    20 08                   JR      NZ,ERROR        ;YES,ERROR
2031    ED 78                   IN      A,(C)           ;READ LOOP BACKED DATA
2033    B8                      CP      B               ;DATA OK ?
2034    20 0E                   JR      NZ,DERROR       ;NO,DERROR
2036    04                      INC     B               ;UPDATE DATA
2037    18 E0                   JR      XMIT            ;GO AND TEST NEXT DATA
                        ;
2039    CB 77           ERROR:  BIT     6,A             ;FRAMING ERROR ?
203B    20 05                   JR      NZ,FERR         ;YES,FERR
203D    CB 6F                   BIT     5,A             ;RX OVERRUN ERROR ?
203F    20 02                   JR      NZ,OERR         ;YES,OERR
                        ;       PARITY ERROR DETECTED
2041    76              PERR:   HALT                    ;
                        ;       FRAMING ERROR DETECTED
2042    76              FERR:   HALT                    ;
                        ;       OVERRUN ERROR DETECTED
2043    76              OERR:   HALT                    ;
                        ;
```

（次ページへつづく）

```
                          ;       DATA ERROR DETECTED
2044    76                DERROR: HALT                        ;
                          ;
                          ;       SIO CHANNEL A COMMAND CHAIN
2045    18                SIOACC: DB      18H     ;WR0    CHANNEL RESET
2046    01                        DB      01H     ;WR0    POINTS WR1
2047    00                        DB      00H     ;WR1    DISABLE ALL INTERRUPTS
2048    03                        DB      03H     ;WR0    POINTS WR3
2049    C1                        DB      0C1H    ;WR3    RX 8 BITS/CHARACTER
                                                  ;       RX ENABLE
204A    04                        DB      04H     ;WR0    POINTS WR4
204B    47                        DB      47H     ;WR4    X16 CLOCK
                                                  ;       1 STOP BIT
                                                  ;       EVEN PARITY
                                                  ;       ENABLE PARITY
204C    05                        DB      05H     ;WR0    POINTS WR5
204D    EA                        DB      0EAH    ;WR5    DTR ON
                                                  ;       TX 8 BITS/CHARACTER
                                                  ;       TX ENABLE
                                                  ;       RTS ON
                          ;
204E                      SIOACL  EQU     $               ;END OF SIOACC
                          ;
204E                              DS      64              ;STACK AREA
208E                      STACK   EQU     $               ;
                          ;
                          ;
                                  END
```

プログラムはまずSIOのRR0を読み，トランスミッタ・バッファ・エンプティになるのを待ち，そうであれば(RR0のビット2＝"1")レジスタB中のデータを送信します．

次にプログラムはSIOのRR0を読み，レシーバにデータが用意されるのを待ちます．レシーバにデータがそろう（RR0のビット0＝"1"）と，RR1を読み取り，エラーが発生したか否かをチェックします．エラーがあれば（RR0のビット4～6）エラー・ルーチンへとびエラーの詳細をチェックします．

エラーがなければ，受信データ・レジスタからデータを読み取り，それが送信したデータと一致することをチェックし，一致していればデータを1だけ進めて，再度データを送信しに行きます．

図44にProg❷における，SIOのイニシャライゼーション・コマンド・チェインを示します．このコマンド・チェインではWR3に対してコマンドを書き込むのと同時にレシーバをイネーブルしていますが，一般的にはレシーバに対するコマンドをすべて出してから，それをイネーブルするのが安全なやりかたであり，こ

<図44>
Prog❷ のイニシャライゼーション・コマンド・チェイン（非同期モード，ポーリング）

	レジスタ	コマンド（16進数）	コマンドのビット・パターン 7 6 5 4 3 2 1 0	説明
チャネル	WR0	18	0 0 0 1 1 0 0 0	リセット・チャネル
	WR0	01	0 0 0 0 0 0 0 1	WR1をポイント
	WR1	00	0 0 0 0 0 0 0 0	割り込みをディスエーブル
	WR0	03	0 0 0 0 0 0 1 1	WR3をポイント
	WR3	C1	1 1 0 0 0 0 0 1	受信8ビット/キャラクタ レシーバ・イネーブル
	WR0	04	0 0 0 0 0 1 0 0	WR4をポイント
	WR4	47	0 1 0 0 0 1 1 1	クロック倍率＝16 ストップ・ビット＝1ビット 偶数パリティ イネーブル・パリティ
	WR0	05	0 0 0 0 0 1 0 1	WR5をポイント
	WR5	EA	1 1 1 0 1 0 1 0	DTR ON 送信8ビット/キャラクタ トランスミッタ・イネーブル RTS ON

	レジスタ	コマンド（16進数）	コマンドのビット・パターン 7 6 5 4 3 2 1 0	説明
チャネルA	WR0	18	0 0 0 1 1 0 0 0	リセット・チャネル
	WR0	01	0 0 0 0 0 0 0 1	WR1をポイント
	WR1	13	0 0 0 1 0 0 1 1	受信キャラクタ割り込み可 送信割り込み可 外部/ステータス割り込み可
	WR0	03	0 0 0 0 0 0 1 1	WR3をポイント
	WR3	C1	1 1 0 0 0 0 0 1	受信8ビット/キャラクタ レシーバ・イネーブル
	WR0	04	0 0 0 0 0 1 0 0	WR4をポイント
	WR4	47	0 1 0 0 0 1 1 1	クロック倍率＝16 ストップ・ビット＝1ビット 偶数パリティ イネーブル・パリティ
	WR0	05	0 0 0 0 0 1 0 1	WR5をポイント
	WR5	EA	1 1 1 0 1 0 1 0	DTR ON 送信8ビット/キャラクタ トランスミッタ・イネーブル RTS ON
チャネルB	WR0	18	0 0 0 1 1 0 0 0	リセット・チャネル
	WR0	01	0 0 0 0 0 0 0 1	WR1をポイント
	WR1	04	0 0 0 0 0 1 0 0	ステータス・アフェクツ・ベクトル
	WR0	02	0 0 0 0 0 0 1 0	WR2をポイント
	WR2	××	××××××××	割り込みベクトル

〈図45〉
Prog❸ のイニシャライゼーション・コマンド・チェイン（非同期モード，割り込み駆動型）

のコマンド・チェインでは WR4 以降にレシーバをイネーブルするのが正しいやりかたです．

このことはこれ以降に示す，すべてのプログラムに対していえます．ただし，本項で示す，ループ・バック・テストに関する限り，WR3 に対するコマンドを書き込むのと同時にレシーバをイネーブルしたとしてもなんらの支障はありません．

ここで示すすべてのプログラムの動作は，筆者の自作の Z80 コンピュータにおいて検証済みです．

SIO ループ・バック・テスト（非同期モード，割り込み駆動型）

Prog❸ に割り込みを使用した非同期モードのループ・バック・テストを示します．このプログラムでは，Z80 CPU のモード2割り込みを使用しており，かつ SIOではステータス・アフェクツ・ベクトル機能を用いています．

図45に **Prog❸** における SIO のイニシャライゼーション・コマンド・チェインを示します．このコマンド・チェインにおいて，**図44**のコマンド・チェイン（非同期モード，ポーリング）と異なる点は，チャネルAの WR3 に対するコマンドと，割り込みベクトルをロードするために，チャネルBに対してもコマンドを送る必要がある点です．

このプログラムの動作は次のとおりです．プログラムはまず Z80 CPU のインタラプト・レジスタをセットし，ついでモード2割り込みを使うことを宣言します．次にボーレート・ジェネレータをセットアップし，SIOのチャネルAおよびチャネルBに対して，コマンド・チェインをロードし，データをセットアップした後に CPU の割り込みをイネーブルし，永久ループに入ります．

この後最初に何が起こるかというと，外部／ステータス割り込みが発生します．この理由はイニシャライゼーション・コマンド中の WR3 に外部／ステータス割り込み可とセットしており，かつ WR5 により DTR および RTS を ON にしているからです．

図42からわかるように，SIO の $\overline{\mathrm{DTRA}}$ は $\overline{\mathrm{DCDA}}$ に，$\overline{\mathrm{RTSA}}$ は $\overline{\mathrm{CTSA}}$ に接続されていますので，$\overline{\mathrm{DTRA}}$ または $\overline{\mathrm{RTSA}}$ が"L"になると SIO は外部／ステータス割り込みを発生します（SIO はリセットされると，$\overline{\mathrm{DTRA}}$ および $\overline{\mathrm{RTSA}}$ を "H" にする）．

したがって，プログラムのコントロールはまずチャネルAの外部／ステータス割り込みルーチンである I05 にきます．このルーチンでは外部／ステータス割り込みをリセットし，RR0 を読み，CTS が"1"であることを確認した後に，レジスタD中の送信データを出力し，これを1だけ更新します．

〈Prog ❸〉 SIO ループ・バック・テスト（非同期モード，割り込み駆動型）

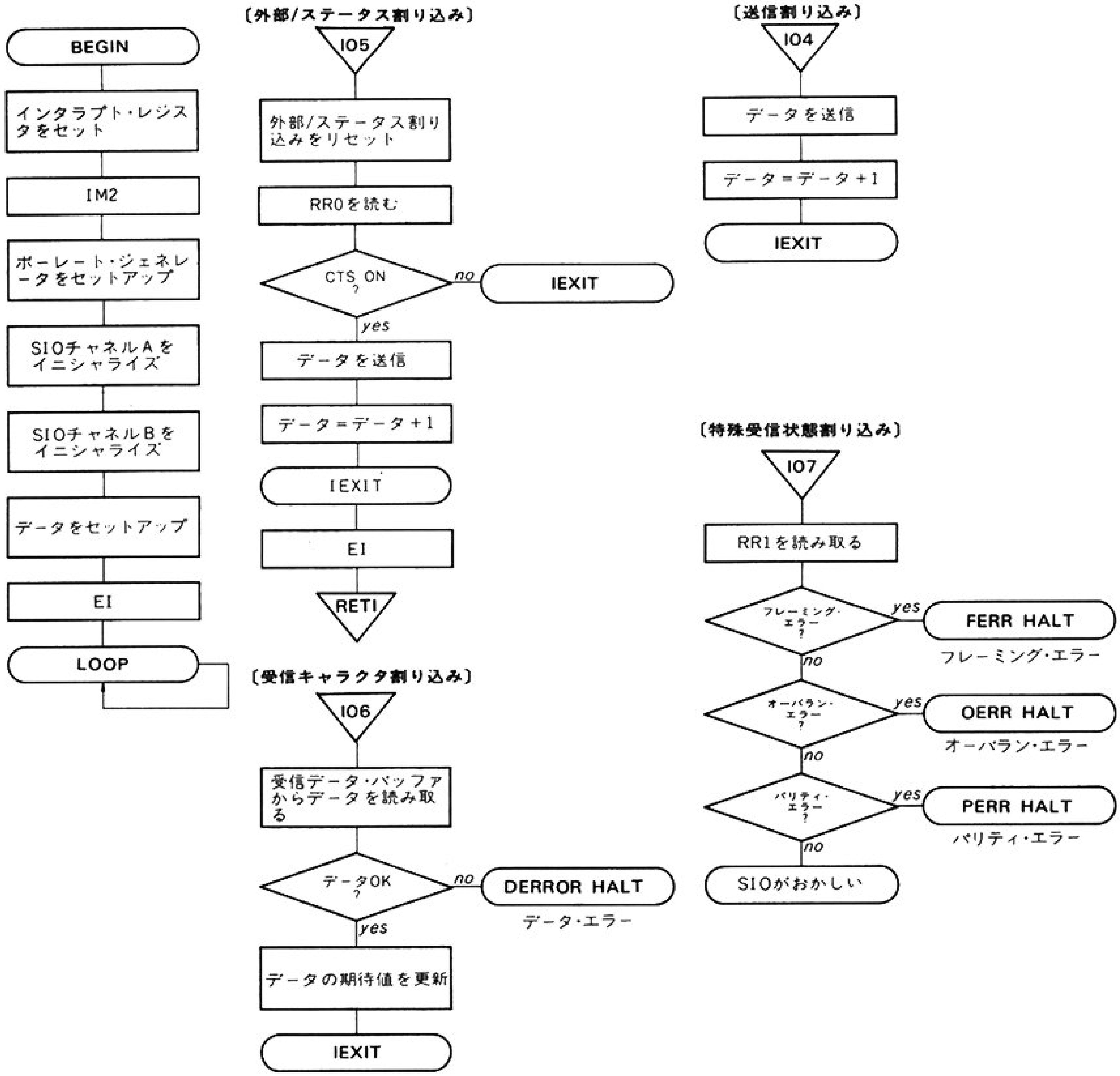
```
                        ;       Z80 SIO LOOP BACK TEST (ASYNC,INTERRUPT DRIVEN)
                        ;
                        ;       THIS PROGRAM TRANSMITS DATA TO SIO CHANNEL A
                        ;       AND RECEIVES LOOP-BACKED DATA FROM THE SAME
                        ;       CHANNEL.
                        ;
0009                    CTC1    EQU     09H             ;CTC CHANNEL 1
                        ;
0004                    SIOAD   EQU     04H             ;SIO CHANNEL A DATA
0005                    SIOAC   EQU     SIOAD+1         ;SIO CHANNEL A CONTROL
0006                    SIOBD   EQU     SIOAC+1         ;SIO CHANNEL B DATA
0007                    SIOBC   EQU     SIOBD+1         ;SIO CHANNEL B CONTROL
                        ;
0000                            ASEG
                                ORG     2000H
2000    F3              BEGIN:  DI                      ;DISABLE INTERRUPT
2001    31 20AB                 LD      SP,STACK        ;SET STACK POINTER
2004    3E 20                   LD      A,20H           ;SET INTERRUPT REGISTER
2006    ED 47                   LD      I,A             ;
2008    ED 5E                   IM      2               ;SET MODE 2 INTERRUPT
                        ;       SET UP BAUD RATE GENERATOR
200A    3E 47                   LD      A,47H           ;MODE WORD FOR CTC
200C    D3 09                   OUT     (CTC1),A        ;LOAD MODE WORD TO CTC
200E    3E 0A                   LD      A,10            ;SET TIME CONSTANT
2010    D3 09                   OUT     (CTC1),A        ;LOAD TIME CONSTANT
                        ;       INITIALIZE SIO CHANNEL A
2012    21 205D                 LD      HL,SIOACC       ;LOAD COMMAND ADDRESS
2015    06 09                   LD      B,SIOACL-SIOACC ;LOAD COMMAND LENGTH
2017    0E 05                   LD      C,SIOAC         ;LOAD SIO A CTL ADRS
2019    ED B3                   OTIR                    ;LOAD COMMAND TO SIO A
                        ;       INITIALIZE SIO CHANNEL B FOR INTERRUPT
201B    06 05                   LD      B,SIOBCL-SIOBCC ;LOAD COMMAND LENGTH
201D    0E 07                   LD      C,SIOBC         ;LOAD SIO B CTL ADRS
                        ;       NOTE THAT HL POINTS SIOBCC NOW.
201F    ED B3                   OTIR                    ;LOAD COMMAND TO SIO B
                        ;       SET UP DATA AND SIO A DATA ADDRESS
                        ;       D=DATA TO BE TRANSMITTED
                        ;       E=DATA TO BE EXPECTED
                        ;       C=SIO CHANNEL A DATA ADDRESS
2021    11 0000                 LD      DE,0            ;SET UP DATA
2024    0E 04                   LD      C,SIOAD         ;LOAD SIO A DATA ADRS
                        ;       NOW TEST BEGINS
2026    FB                      EI                      ;ENABLE INTERRUPT
2027    18 FE                   JR      $               ;LOOOP ETERNALLY
                        ;
                        ;       CHANNEL A TRANSMITER BUFFER EMPTY INTERRUPT
2029    ED 51           I04:    OUT     (C),D           ;TRANSMIT DATA
202B    14                      INC     D               ;UPDATE DATA
                        ;       RE-ENABLE INTERRUP AND RETURN
202C    FB              IEXIT:  EI                      ;RE-ENABLE INTERRUPT
202D    ED 4D                   RETI                    ;
                        ;
                        ;       CHANNEL A EXTERNAL STATUS INTERRUPT
202F    3E 10           I05:    LD      A,10H           ;RESET EXTERNAL INT
2031    D3 05                   OUT     (SIOAC),A       ;
2033    DB 05                   IN      A,(SIOAC)       ;READ RR0
2035    CB 6F                   BIT     5,A             ;CTS ON ?
2037    28 F3                   JR      Z,IEXIT         ;NO,IEXIT
2039    ED 51                   OUT     (C),D           ;TRANSMIT FIRST DATA
203B    14                      INC     D               ;UPDATE DATA
203C    18 EE                   JR      IEXIT           ;
                        ;
```

```
                         ;       CHANNEL A RECEIVE CHARACTER AVAILABLE INTERRUPT
203E    ED 78            I06:    IN      A,(C)               ;READ DATA FROM SIO
2040    BB                       CP      E                   ;DATA OK ?
2041    20 03                    JR      NZ,DERROR           ;NO,DERROR
2043    1C                       INC     E                   ;UPDATE DATA EXPECTED
2044    18 E6                    JR      IEXIT               ;
                         ;
                         ;       DATA ERROR
2046    76               DERROR: HALT                        ;
                         ;
                         ;       CHANNEL A SPECIAL RECEIVE CONDITION
2047    3E 01            I07:    LD      A,1                 ;TO ACCESS RR1
2049    D3 05                    OUT     (SIOAC),A           ;SELECT RR1
204B    DB 05                    IN      A,(SIOAC)           ;READ RR1
204D    CB 77                    BIT     6,A                 ;FRAMING ERROR ?
204F    20 09                    JR      NZ,FERR             ;YES,FERR
2051    CB 6F                    BIT     5,A                 ;RX OVERRUN ?
2053    20 06                    JR      NZ,OERR             ;YES,OERR
2055    CB 67                    BIT     4,A                 ;PARITY ERORR ?
2057    20 03                    JR      NZ,PERR             ;YES,PERR
                         ;       SIO ERROR
2059    76                       HALT                        ;SIO ERROR
                         ;       FRAMING ERROR
205A    76               FERR:   HALT                        ;
                         ;       RX OVERRUN ERROR
205B    76               OERR:   HALT                        ;
                         ;       PARITY ERROR
205C    76               PERR:   HALT                        ;
                         ;
                         ;       SIO CHANNEL A COMMAND CHAIN
205D    18               SIOACC: DB      18H      ;WR0    CHANNEL RESET
205E    01                       DB      01H      ;WR0    POINTS WR1
205F    13                       DB      13H      ;WR1    INT ON ALL RX CHRACTERS
                                                  ;       TX INT ENABLE
                                                  ;       EXT INT ENABLE
2060    03                       DB      03H      ;WR0    POINTS WR3
2061    C1                       DB      0C1H     ;WR3    RX 8 BITS/CHARACTER
                                                  ;       RX ENABLE
2062    04                       DB      04H      ;WR0    POINTS WR4
2063    47                       DB      47H      ;WR4    X16 CLOCK
                                                  ;       1 STOP BIT
                                                  ;       PARITY EVEN
                                                  ;       PARITY ENABLE
2064    05                       DB      05H      ;WR0    POINTS WR5
2065    EA                       DB      0EAH     ;WR5    DTR ON
                                                  ;       TX 8 BITS/CHARACTER
                                                  ;       TX ENABLE
                                                  ;       RTS ON
                         ;
2066                     SIOACL  EQU     $                   ;END OF SIOACC
                         ;
                         ;       SIO CHANNEL B COMMAND CHAIN
2066    18               SIOBCC: DB      018H     ;WR0    CHANNEL RESET
2067    01                       DB      01H      ;WR0    POINTS WR1
2068    04                       DB      04H      ;WR1    STATUS AFFECTS VECTOR
2069    02                       DB      02H      ;WR0    POINTS WR2
206A    F0                       DB INTV-BEGIN    ;WR2    INTERRUPT VECTOR
                         ;
206B                     SIOBCL  EQU     $                   ;END OF SIOBCC
                         ;
                         ;
206B                             DS      64                  ;STACK AREA
20AB                     STACK   EQU     $                   ;
                         ;
```

(次ページへつづく)

```
                              ORG     BEGIN+0F0H
                        ;     INTERRUPT VECTOR
20F0    0000            INTV: DW      0          ;CH B TX BUFFER EMPTY
20F2    0000                  DW      0          ;CH B EXTERNAL STATUS
20F4    0000                  DW      0          ;CH B RX AVAILABLE
20F6    0000                  DW      0          ;CH B SPECIAL RECV COND.
20F8    2029                  DW      I04        ;CH A TX BUFFER EMPTY
20FA    202F                  DW      I05        ;CH A EXTERNAL STATUS
20FC    203E                  DW      I06        ;CH A RX AVAILABLE
20FE    2047                  DW      I07        ;CH A SPECIAL RECV COND.
                        ;
                        ;
                              END
```

〈図46〉 Prog ❸ のフローチャート（非同期モード，割り込み駆動型）



<Prog ⓬> SIO ループ・バック・テスト（同期モード，割り込み駆動型）

```
                        ;       Z80 SIO LOOP BACK TEST (SYNC, INTERRUPT DRIVEN)
                        ;
                        ;       THIS PROGRAM TRANSMITS DATA TO SIO CHANNEL A
                        ;       IN SYNCHRONOUS MODE AND RECEIVES DATA FROM THE
                        ;       SAME CHANNEL.
                        ;
0009                    CTC1    EQU     09H                 ;CTC CHANNEL 1
                        ;
0004                    SIOAD   EQU     04H                 ;SIO CHANNEL A DATA
0005                    SIOAC   EQU     SIOAD+1             ;SIO CHANNEL A CONTROL
0006                    SIOBD   EQU     SIOAC+1             ;SIO CHANNEL B DATA
0007                    SIOBC   EQU     SIOBD+1             ;SIO CHANNEL B CONTROL
                        ;
0016                    SYNC    EQU     16H                 ;SYNC CHARACTER
                        ;
2400                    BUFFER  EQU     2400H               ;RECV. CHAR. BUFFER
                        ;
0000                            ASEG
                                ORG     2000H
2000    F3              BEGIN:  DI                          ;DISABLE INTERRUPT
2001    31 2100                 LD      SP,STACK            ;SET STACK POINTER
2004    3E 21                   LD      A,21H               ;SET INTERRUPT REGISTER
2006    ED 47                   LD      I,A                 ;
2008    ED 5E                   IM      2                   ;SET MODE 2 INTERRUPT
                        ;       SET UP BAUD RATE GENERATOR
200A    3E 47                   LD      A,47H               ;MODE WORD FOR CTC
200C    D3 09                   OUT     (CTC1),A            ;LOAD MODE WORD TO CTC1
200E    3E A0                   LD      A,160               ;SET TIME CONSTANT
                                                            ;9600 BAUD
2010    D3 09                   OUT     (CTC1),A            ;LOAD TIME CONSTANT
                        ;       INITIALIZE SIO CHANNEL A
2012    21 20AE                 LD      HL,SIOACC           ;LOAD COMMAND ADDRESS
2015    06 0D                   LD      B,SIOACL-SIOACC     ;LOAD COMMAND LENGTH
2017    0E 05                   LD      C,SIOAC             ;LOAD SIOA CTL ADRS
2019    ED B3                   OTIR                        ;LOAD COMMAND TO SIO
                        ;
                        ;       INITIALIZE SIO CHANNEL B FOR INTERRUPT
201B    06 05                   LD      B,SIOBCL-SIOBCC     ;LOAD COMMAND LENGTH
201D    0E 07                   LD      C,SIOBC             ;LOAD SIO B CTL ADRS
                        ;       NOTE THAT HL POINTS SIOBCC
201F    ED B3                   OTIR                        ;LOAD COMMAND TO SIO B
                        ;       SET UP DATA AND CONTROL FLAG
                        ;       D=DATA TO BE TRANSMITTED
                        ;       E=DATA TO BE EXPECTED
                        ;       B=CONTROL FLAG
                        ;       BIT 0   ONE CYCLE END
                        ;       BIT 1   FIRST CHARACTER TRANSMITTED
                        ;       BIT 2   SYNCHRONIZED
2021    11 0000         BEGIN0: LD      DE,0                ;SET UP DATA
2024    06 00                   LD      B,0                 ;RESET CONTROL FLAG
                        ;       NOW TEST BEGINS
2026    21 2400                 LD      HL,BUFFER           ;LOAD BUFFER ADDRESS
2029    FB                      EI                          ;ENABLE INTERRUPT
202A    C3 202A                 JP      $                   ;LOOP ETERNALLY
                        ;
                        ;       CHANNEL A TRANSMITTER BUFFER EMPTY INTERRUPT
202D    CB 40           I04:    BIT     0,B                 ;ONE CYCLE END ?
202F    C2 2051                 JP      NZ,I041             ;YES,I041
2032    CB 48                   BIT     1,B                 ;FIRST CHARACTER ?
2034    C2 203D                 JP      NZ,I040             ;NO,I040
2037    CB C8                   SET     1,B                 ;SET FIRST CHARACTER
                                                            ;TRANSMITTED
2039    3E 80                   LD      A,80H               ;RESET TX CRC GEN.
203B    D3 05                   OUT     (SIOAC),A           ;
203D    7A              I040:   LD      A,D                 ;LOAD DATA
```

```
203E    D3 04           OUT   (SIOAD),A     ;TRANSMIT IT
2040    3E 05           LD    A,05H         ;TO ACCESS WR5
2042    D3 05           OUT   (SIOAC),A     ;
2044    3E EF           LD    A,0EFH        ;ENABLE TX CRC
2046    D3 05           OUT   (SIOAC),A     ;
2048    14              INC   D             ;UPDATE DATA
2049    C2 204E         JP    NZ,IEXIT      ;ONE CYCLE END ?
                                            ;NO,IEXIT
204C    CB C0           SET   0,B           ;SET ONE CYCLE END
                ;       EXIT FROM INTERRUPT
204E    FB      IEXIT:  EI                  ;RE-ENABLE INTERRUPT
204F    ED 4D           RETI                ;
                ;
2051    3E 05   I041:   LD    A,05H         ;TO ACCESS WR5
2053    D3 05           OUT   (SIOAC),A     ;
2055    3E E0           LD    A,0E0H        ;DISABLE TRANSMITTER
                                            ;DROP RTS
2057    D3 05           OUT   (SIOAC),A     ;
2059    C3 204E         JP    IEXIT         ;
                ;
                ;       EXTERNAL STATUS INTERRUPT
205C    DB 05   I05:    IN    A,(SIOAC)     ;READ RR0
205E    4F              LD    C,A           ;SAVE STATUS FOR LATER
205F    3E 10           LD    A,10H         ;RESET EXTERNAL INT
2061    D3 05           OUT   (SIOAC),A     ;
2063    79              LD    A,C           ;LOAD STATUS
2064    CB 67           BIT   4,A           ;SYNCHRONIZED ?
2066    CA 2078         JP    Z,I050        ;NO,I050
2069    CB 50           BIT   2,B           ;ALREADY IN SYNC ?
206B    C2 2078         JP    NZ,I050       ;YES,I050
206E    CB D0           SET   2,B           ;SET IN SYNC
2070    3E 03           LD    A,03H         ;TO ACCESS WR3
2072    D3 05           OUT   (SIOAC),A     ;
2074    3E C1           LD    A,0C1H        ;DROP HUNT MODE
                                            ;ENABLE SYNC LOAD
2076    D3 05           OUT   (SIOAC),A     ;
2078    79      I050:   LD    A,C           ;RELOAD STATUS
2079    CB 67           BIT   4,A           ;CTS ON ?
207B    CA 204E         JP    Z,IEXIT       ;NO,IEXIT
207E    CB 48           BIT   1,B           ;FIRST CHAR. XMITTED ?
2080    C2 204E         JP    NZ,IEXIT      ;YES,IEXIT
2083    C3 202D         JP    I04           ;GO AND TRANSMIT FIRST
                                            ;CHARACTER
                ;
                ;       RECEIVE CHARACTER AVAILABLE INTERRUPT
2086    DB 04   I06:    IN    A,(SIOAD)     ;READ DATA FROM SIO
2088    77              LD    (HL),A        ;STORE DATA INTO BUF.
2089    23              INC   HL            ;BUMP POINTER
208A    CB 50           BIT   2,B           ;IN SYNC ?
208C    CA 204E         JP    Z,IEXIT       ;NO,IEXIT
208F    BB              CP    E             ;DATA OK ?
2090    C2 20AB         JP    NZ,DERROR     ;NO,DERROR
2093    1C              INC   E             ;UPDATE DATA
2094    C2 204E         JP    NZ,IEXIT      ;ONE CYCLE END ?
                                            ;NO,IEXIT
                ;       END OF TEST
2097    C3 2000         JP    BEGIN         ;RE-DO FROM START
                ;
                ;       SPECIAL RECEIVE CONDITION INTERRUPT
209A    3E 01   I07:    LD    A,01H         ;TO ACCESS RR1
209C    D3 05           OUT   (SIOAC),A     ;
209E    DB 05           IN    A,(SIOAC)     ;READ RR1
20A0    CB 6F           BIT   5,A           ;RX OVERRUN ERROR ?
20A2    C2 20AC         JP    NZ,OERR       ;YES,OERR
20A5    CB 67           BIT   4,A           ;PARITY ERROR ?
20A7    C2 20AD         JP    NZ,PERR       ;YES,PERR
```

```
                        ;       SIO ERROR
20AA    76                      HALT                            ;
                        ;       DATA ERROR
20AB    76              DERROR: HALT                            ;
                        ;       OVERRUN ERROR
20AC    76              OERR:   HALT                            ;
                        ;       PARITY ERROR
20AD    76              PERR:   HALT                            ;
                        ;
                        ;       SIO CHANNEL A COMMAND CHAIN
20AE    18              SIOACC: DB      18H     ;WR0    RESET CHANNEL
20AF    01                      DB      01H     ;WR0    POINTS WR1
20B0    13                      DB      13H     ;WR1    INT ON ALL RX CHAR.S
                                                ;       TX INT ENABLE
                                                ;       EXT INT ENABLE
20B1    03                      DB      03H     ;WR0    POINTS WR3
20B2    D3                      DB      0D3H    ;WR3    RX 8BITS/CHARACTER
                                                ;       ENTER HUNT PHASE
                                                ;       SYNC LOAD INHIBIT
                                                ;       RX ENABLE
20B3    04                      DB      04H     ;WR0    POINTS WR4
20B4    13                      DB      13H     ;WR4    X1 CLOCK MODE
                                                ;       16 BITS SYNC CHAR.
                                                ;       SYNC MODE
                                                ;       PARITY EVEN
                                                ;       ENABLE PARITY
20B5    06                      DB      06H     ;WR0    POINTS WR6
20B6    16                      DB      SYNC    ;WR6    SYNC CHARACTER
20B7    07                      DB      07H     ;WR0    POINTS WR7
20B8    16                      DB      SYNC    ;WR7    SYNC CHARACTER
20B9    05                      DB      05H     ;WR0    POINTS WR5
20BA    EE                      DB      0EEH    ;WR5    DTR ON
                                                ;       TX 8 BITS/CHARACTER
                                                ;       TX ENABLE
                                                ;       RTS ON
                        ;
20BB                    SIOACL  EQU     $               ;END OF SIOACC
                        ;
                        ;       SIO CHANNEL B COMMAND CHAIN
20BB    18              SIOBCC: DB      018H    ;WR0    RESET CHANNEL
20BC    01                      DB      01H     ;WR0    POINTS WR1
20BD    04                      DB      04H     ;WR1    STATUS AFFECTS VECTOR
20BE    02                      DB      02H     ;WR0    POINTS WR2
20BF    F0                      DB      0F0H    ;WR2    INTERRUPT VECTOR
                        ;
20C0                    SIOBCL  EQU     $               ;END OF SIOBCC
                        ;
20C0                            DS      64              ;STACK AREA
2100                    STACK   EQU     $               ;STACK FRAME
                        ;
                                ORG     21F0H
21F0    0000            INTV:   DW      0               ;CH B TX BUFFER EMPTY
21F2    0000                    DW      0               ;CH B EXTERNAL INT
21F4    0000                    DW      0               ;CH B RX CHAR. AVAIL.
21F6    0000                    DW      0               ;CH B SPECIAL RECV COND.
21F8    202D                    DW      I04             ;CH A TX BUFFER EMPTY
21FA    205C                    DW      I05             ;CH A EXTERNAL INT
21FC    2086                    DW      I06             ;CH A RX CHAR. AVAIL.
21FE    209A                    DW      I07             ;CH B SPECIAL RECV COND.
                        ;
                        ;
                                END
```

ループ・バックしたデータがレシーバに入ると，チャネルAの受信キャラクタ割り込みが発生し，プログラムのコントロールは受信キャラクタ割り込みルーチンであるI06にきます．このI06ルーチンでは受信データ・バッファからデータを読み取り，それが期待すべきもの（レジスタE）であるか否かをチェックし，正しいデータであれば期待すべきデータを1だけ更新します．

送信バッファが「空」になると，チャネルAのトランスミッタ・バッファ・エンプティ割り込みが発生し，プログラムのコントロールはその割り込みルーチンであるI04にきます．このI04ルーチンでは次のデータを送信し，送信データを更新します．

受信キャラクタに何らかのエラーが検出されると，チャネルAの特殊受信状態割り込みが発生し，プログラムのコントロールは，その割り込みルーチンであるI07にきます．このルーチンではRR1を読み取り，エラーの原因（フレーミング・エラー，オーバラン・エラー，パリティ・エラー）の識別を行います．図46にProg❸のフロー・チャートを示します(304ページ)．

SIOループ・バック・テスト（同期モード，割り込み駆動型）

Prog❹に同期モードでかつ割り込み駆動型のループ・バック・テストを示します．このプログラムではSIOは同期モードで動作しますので，送・受信クロッ

〈図47〉
Prog❹のイニシャライゼーション・コマンド・チェイン（同期モード，割り込み駆動型）

	レジスタ	コマンド（16進数）	コマンドのビット・パターン 7 6 5 4 3 2 1 0	説明
チャネルA	WR0	18	0 0 0 1 1 0 0 0	リセット・チャネル
	WR0	01	0 0 0 0 0 0 0 1	WR1をポイント
	WR1	13	0 0 0 1 0 0 1 1	受信キャラクタ割り込み可 送信割り込み可 外部/ステータス割り込み可
	WR0	03	0 0 0 0 0 0 1 1	WR3をポイント
	WR3	D3	1 1 0 1 0 0 1 1	受信8ビット/キャラクタ エンター・ハント・モード シンク・キャラクタのロードを禁止 レシーバ・イネーブル
	WR0	04	0 0 0 0 0 1 0 0	WR4をポイント
	WR4	13	0 0 0 1 0 0 1 1	クロック倍率＝1 16ビット・シンク・キャラクタ 同期モード 偶数パリティ パリティ・イネーブル
	WR0	06	0 0 0 0 0 1 1 0	WR6をポイント
	WR6	16	0 0 0 1 0 1 1 0	シンク・キャラクタ1
	WR0	07	0 0 0 0 0 1 1 1	WR7をポイント
	WR7	16	0 0 0 1 0 1 1 0	シンク・キャラクタ2
	WR0	05	0 0 0 0 0 1 0 1	WR5をポイント
	WR5	EE	1 1 1 0 1 1 1 0	DTR　ON 送信8ビット/キャラクタ トランスミッタ・イネーブル RTS　ON
チャネルB	WR0	18	0 0 0 1 1 0 0 0	リセット・チャネル
	WR0	01	0 0 0 0 0 0 0 1	WR1をポイント
	WR1	04	0 0 0 0 0 1 0 0	ステータス・アフェクツ・ベクトル
	WR0	02	0 0 0 0 0 0 1 0	WR2をポイント
	WR2	××	× × × × × × × ×	割り込みベクトル

クは1倍でなければなりません.

したがってボーレート・ジェネレータのCTCにロードするタイム・コンスタントはProg❷および❸の場合の16倍(160)にして,9600Hzの出力を出すようにしてあります.またシンク・キャラクタはダブル・シンクとし,シンク・キャラクタ1,シンク・キャラクタ2として16_{16}を用いています.

プログラムはまず,CPUのインタラプト・レジスタをセットし,モード2割り込みを宣言し,ボーレート・ジェネレータをセットアップした後にSIOのチャネルAおよびチャネルBに対してイニシャライゼーション・コマンド・チェインを送ります(図47).

次にプログラムはデータをセットアップし,コントロール・フラグ(レジスタB)をリセットします.このコントロール・フラグは三つの状態を持っています.

① ビット0:1サイクル・エンド
② ビット1:最初のキャラクタを送信した
③ ビット2:同期が確立した

ビット0は256バイトのデータ(00_{16}〜FF_{16})を送信し終わった,という意味です.

次にプログラムはバッファ・ポインタ(HL)をセットし,割り込みをイネーブルし永久ループに入ります.バッファ・ポインタは受信データをバッファ・メモリにストアするためであり,このプログラムのデバッグ用に設けたものです.

最初に起こることはProg❸の場合と同じ理由により,外部/ステータス割り込みが発生します.チャネルAの外部/ステータス割り込みルーチン(I05)では,RR0を読み取りコントロール・フラグに何も立っていなくてかつCTSが"1"のときには,送信割り込みルーチン(I04)にコントロールを移します.

このI04ルーチンではコントロール・フラグに何も立っていないときには最初のデータを送信し,コントロール・フラグのビット1(最初のデータを送信した)をセットし,送信データを更新した後に割り込みをイネーブルし,リターン(RETI)します.

送信バッファが「空」(エンプティ)になると,送信割り込みが発生し,送信割り込みルーチン(I04)が起動されます.このルーチンは次のデータを送信し,データを更新します.256バイトのデータの送信が終わると,このルーチンはコントロール・フラグのビット0を"1"にします.

SIOは最初のデータを送信する前に自動的に2個のシンク・キャラクタを送信します.これを受け取るとレシーバは,同期が確立したという意味の外部/ステータス割り込みを発生します.

外部/ステータス割り込みルーチンではRR0を読み取り,そのビット4(シンク/ハント)を調べ,これが"1"であればコントロール・フラグのビット2(同期が確立した)をセットし,SIOのWR3に対してコマンドを送り,ハント・モードをリセットします.またこのとき同時にシンク・キャラクタのロードをイネーブルします.これによりテスト・データの16_{16}も受信キャラクタとして取り扱われます.

ループ・バックされた送信データがレシーバに入ると,受信キャラクタ割り込みが発生し,プログラムのコントロールは受信キャラクタ割り込みルーチン(I06)にきます.このルーチンでは受信データ・バッファからデータを読み取り,それをバッファにストアした後に,同期の確立がとれているか否かをチェックします(コントロール・フラグのビット2).

コントロール・フラグのビット2が"0"である場合には,受信データを無視し,"1"である場合には受信データが有効であるとみなし,データのチェックを行います.その後データの期待値の更新をしますが,256バイトのデータの受信を終わると,プログラムのコントロールをプログラムの始めに戻し,また始めからテストを行います.このプログラムのフローチャートを図48に示します(次ページ).

SIOループ・バック・テスト (SDLCモード,割り込み駆動型)

Prog❺にSIOのチャネルAを割り込み駆動により,SDLCモードで動作させるためのプログラムを示します.また図49にこのプログラムのSIOイニシャライゼーション・コマンド・チェインを示し,図50にフロー・チャートを示します(316ページ).

プログラムは,まずCPUのインタラプト・レジスタをセットし,モード2割り込みを宣言します.そしてボーレート・ジェネレータをセットアップした後にSIOのチャネルAおよびチャネルBをイニシャライズします.

次にプログラムは,データおよびバッファ・ポインタ,ならびにコントロール・フラグをセットアップします.バッファ・ポインタは同期モードの場合と同様に,プログラムのデバッグのために,受信データをバッファ・メモリにストアするためです.

コントロール・フラグ(レジスタB)は次のような意味を持ちます.

・ビット0:従局アドレスを受信した
・ビット1:256バイトのデータを送信し終えた
・ビット2:CRC受信中
・ビット3:CRC1の受信を終えた(CRC2受信中)

次にプログラムは従局アドレス(55_{16})を送信し,トランスミッタ・アンダラン/EOMラッチをクリヤしま

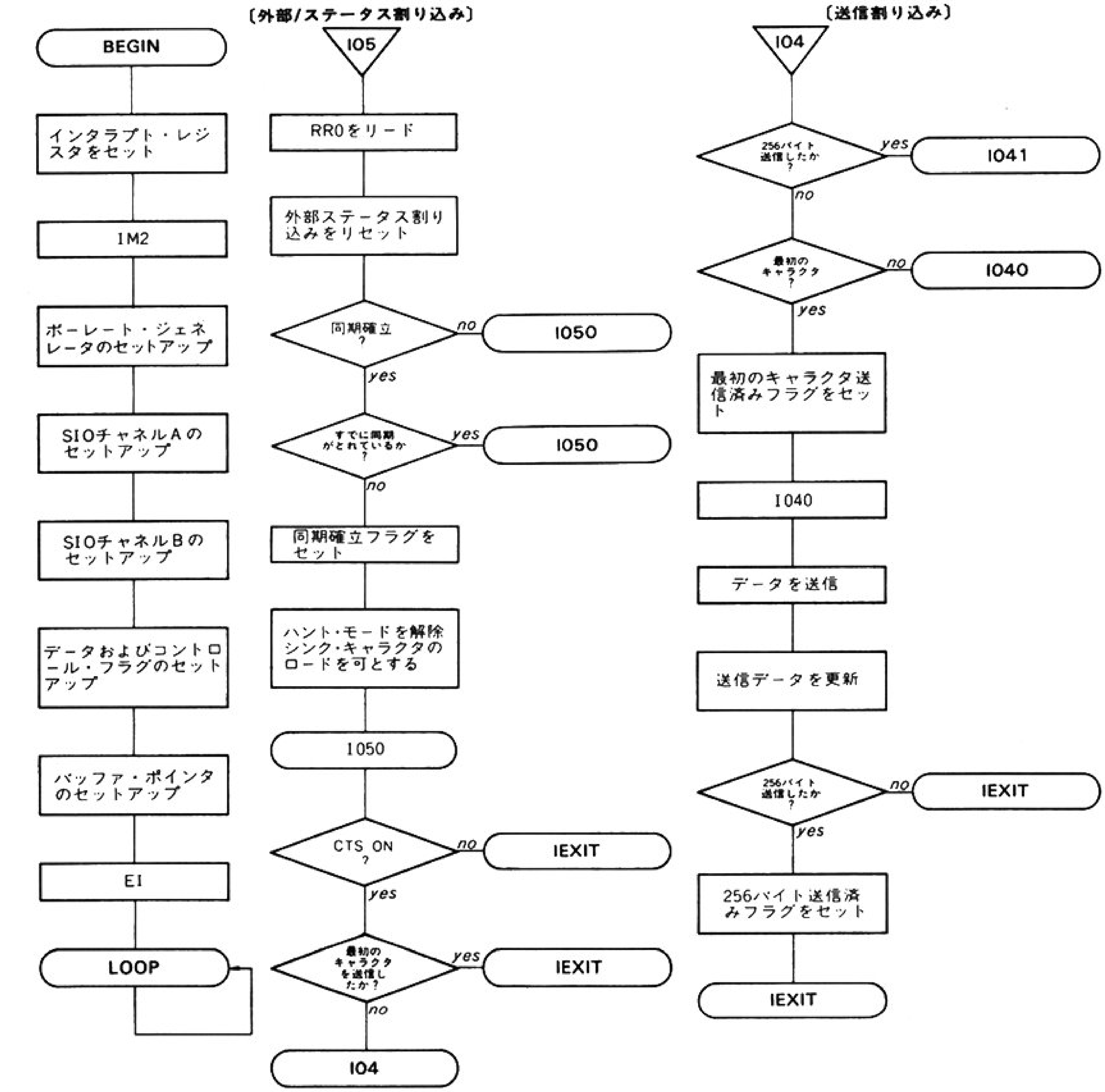
す．これは256バイトのデータを送信し終わり，トランスミッタ・アンダラン状態になったときに，CRCを送信するためです．そしてプログラムは割り込みをイネーブルし，永久ループに入ります．

以上によりSIOはオープニング・フラグに続いて従局アドレスを送信します．レシーバはオープニング・フラグを受信すると，その次のキャラクタ（従局アドレス）がWR6にプログラミングされたものと一致するか否かをチェックし，もし一致していれば，従局アドレスからデータとみなして受信キャラクタ割り込みを発生します．

受信キャラクタ割り込みルーチン（I06）はまずSIOの受信データ・バッファからデータを読み取り，これをバッファにストアします．そしてコントロール・フラグのビット0が立っていなければ，そのデータを従局アドレスであるとしてその値をチェックします．その値がWR6にプログラミングされたものと一致すれば，コントロール・フラグのビット0を“1”にします．

次にトランスミッタ・バッファ・エンプティの割り込みが発生し，送信割り込みルーチンI04が走ります．このルーチンはまずコントロール・フラグのビット1を調べ，これが“1”でない場合にはデータを送信し，かつそのデータの値を更新します．そして256バイト

〈図48〉 Prog❹のフロー・チャート（SIOループ・バック・テスト，



のデータの送信が終わると，コントロール・フラグのビット1をセットします．

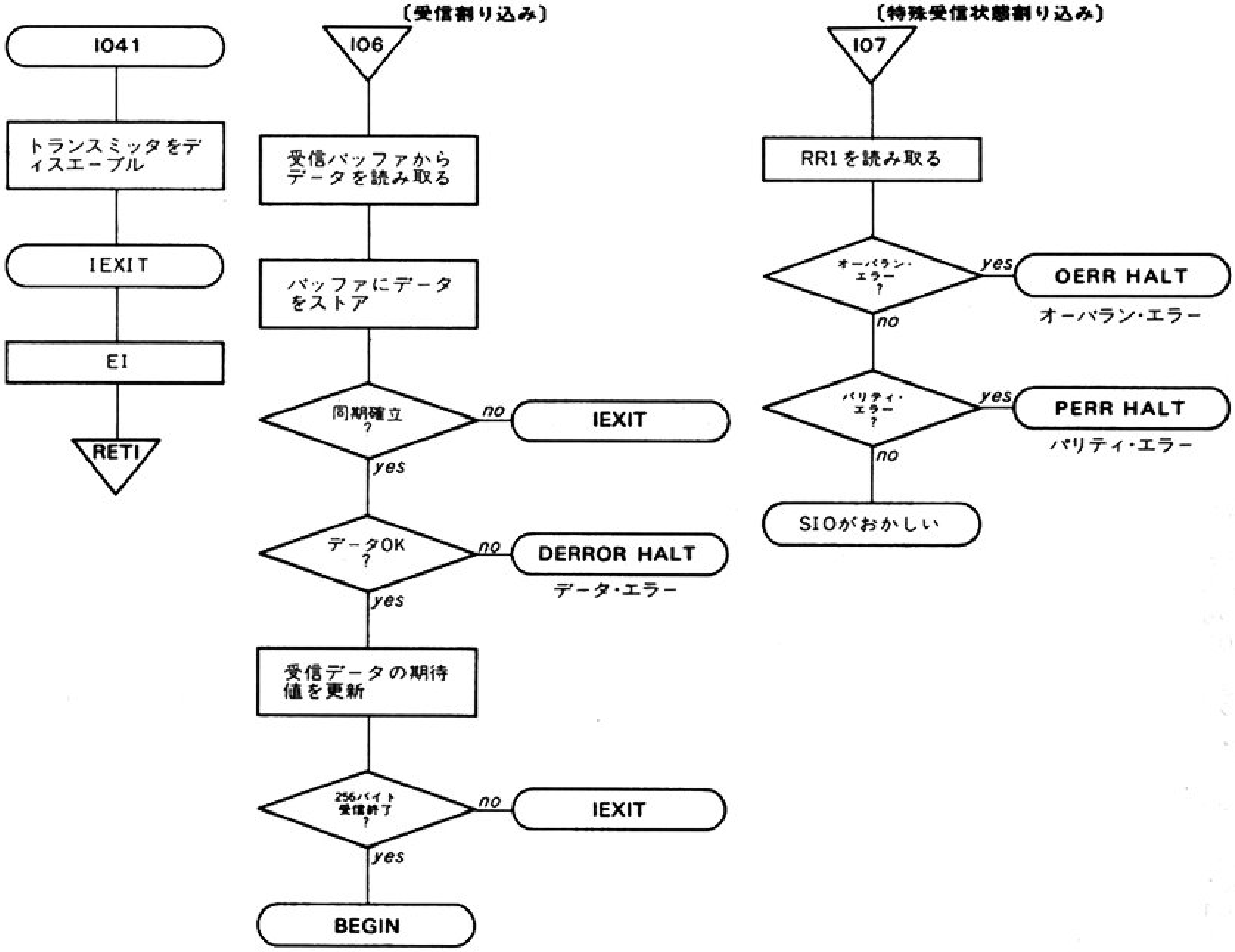
I04ルーチンの始めて，コントロール・フラグのビット1がセットされているときにはCRCおよびクロージング・フラグをSIOに送信させるために，何もしないでリターンします．

受信割り込みルーチン（I06）はコントロール・フラグのビット0（アドレスを受信した）がセットされているときには，受信データが期待すべきものであるか否かチェックし，正しければ期待値を更新します．ここで256バイトのデータの受信が終了すると，次に受信されるのはCRC1ですので，コントロール・フラグのビット2をセットします．

受信割り込みルーチンのはじめにおいて，コントロール・フラグのビット2がセットされていれば，ビット3をチェックし，セットされていなければビット3をセットします．このようにしているうちにCRCの受信が終わり，レシーバがクロージング・フラグを受信すると，特殊受信状態割り込みが発生します．

特殊受信状態割り込みルーチン（I07）はRR1を読み取り，オーバラン・エラーが起きていないことをチェックした後に，RR1のビット7によりエンド・オブ・フレーム割り込みであることを確認し，ついでCRCチェックの結果を調べます（RR1のビット6）．これ

同期モード，割り込み駆動型）



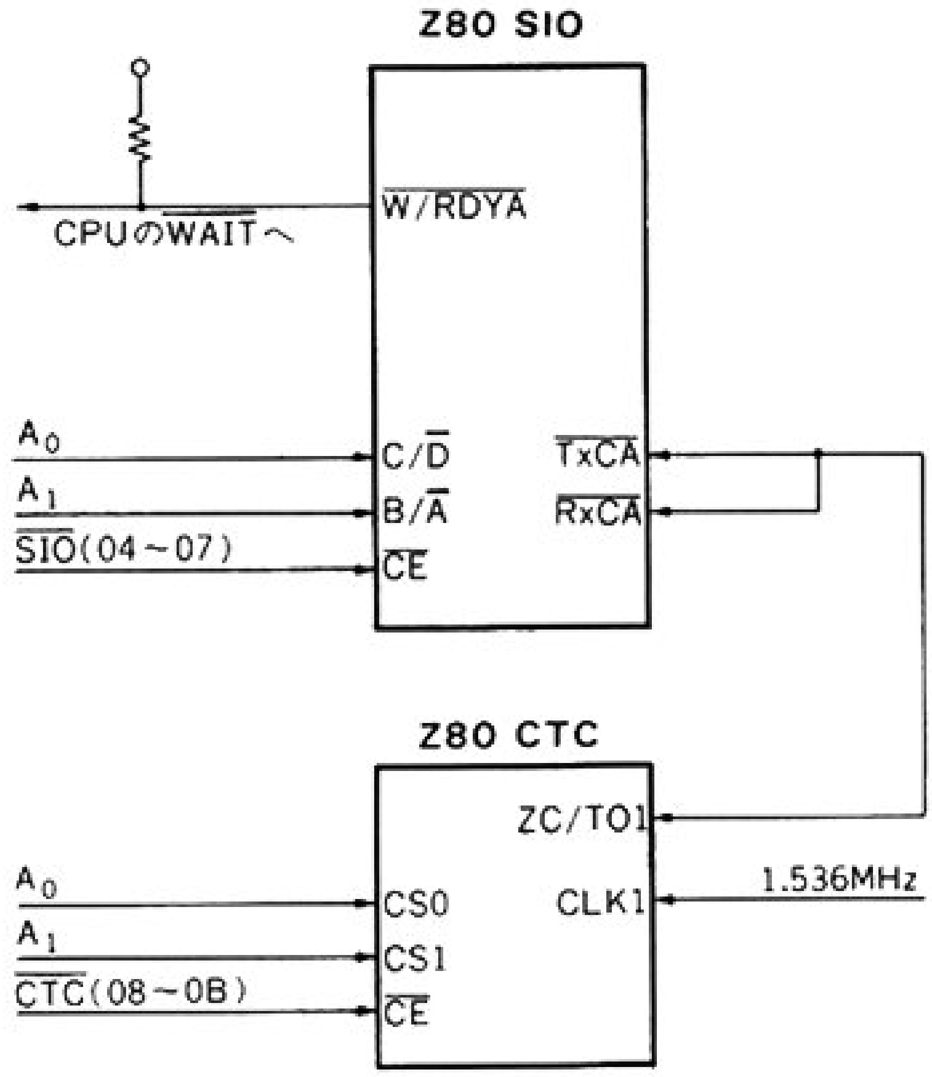
が正しければプログラムは一番最初に戻り，同じテストを繰り返します．

SIOのウェイト機能

SIOには $\overline{\mathrm{W/RDYA(B)}}$ ピンをCPUに対するウェイト信号として使用する機能があります．この機能はWR1のビット7，6，5を適当にプログラミングすることにより達成できます．Prog ❻ にこのSIOのウェイト機能を使って，SIOのチャネルAからデータを送信するプログラムを示します．

またこのプログラムが動作する構成を図51に示します．

〈図51〉 Prog ❻ が動作する構成



SIOのチャネルAに対するイニシャライゼーションは，Prog ❷ のそれとほとんど同じですが，この機能を使用するために WR1 のプログラミングが異なり，かつこのプログラムでは SIO チャネルAのトランスミッタだけしか使用しませんので，レシーバ関係のコマンドはありません．

このプログラムの動作は次のとおりです．まずボーレート・ジェネレータをセットアップし，SIOチャネルAに対してイニシャライゼーション・コマンド・チェインを送ります．そしてバッファに256バイトのデータを用意し，OTIR 命令により SIO のチャネルAに対してそのデータ全部を送信します．OTIR 命令を出

〈Prog ❺〉 SIO ループ・バック・テスト（SDLC，割り込み駆動型）

```
                    ;         Z80 SIO LOOP BACK TEST (SDLC, INTERRUPT DRIVEN)
                    ;
                    ;         THIS PROGRAM TRANSMIT DATA TO SIO CHANNEL A
                    ;         IN SDLC MODE AND RECEIVES DATA FROM THE SAME
                    ;         CHANNEL.
                    ;
0009                CTC1      EQU     09H                ;CTC CHANNEL 1
                    ;
0004                SIOAD     EQU     04H                ;SIO CHANNEL A DATA
0005                SIOAC     EQU     SIOAD+1            ;SIO CHANNEL A CONTROL
0006                SIOBD     EQU     SIOAC+1            ;SIO CHANNEL B DATA
0007                SIOBC     EQU     SIOBD+1            ;SIO CHANNEL B CONTROL
                    ;
0055                ADRS      EQU     55H                ;SECONDARY STATION ADRS
                    ;
2400                BUFFER    EQU     2400H              ;RECV. CHARACTER BUFFER
                    ;
0000                          ASEG
                              ORG     2000H
2000    F3          BEGIN:    DI                         ;DISABLE INTERRUPT
2001    31 20E2               LD      SP,STACK           ;SET STACK POINTER
2004    3E 21                 LD      A,21H              ;SET INTERRUPT REGISTER
2006    ED 47                 LD      I,A                ;
2008    ED 5E                 IM      2                  ;SET MODE 2 INTERRUPT
                    ;         SET UP BAUD RATE GENERATOR
200A    3E 47                 LD      A,47H              ;MODE WORD FOR CTC
200C    D3 09                 OUT     (CTC1),A           ;MODE WORD TO CTC
200E    3E A0                 LD      A,160              ;SET TIME CONSTANT
                                                         ;9600 BAUD
2010    D3 09                 OUT     (CTC1),A           ;LOAD TIME CONSTANT
                    ;         INTIALIZE SIO CHANNEL A
2012    21 2090               LD      HL,SIOACC          ;LOAD COMMAND ADDRESS
2015    06 0D                 LD      B,SIOACL-SIOACC    ;LOAD COMMAND LENGTH
2017    0E 05                 LD      C,SIOAC            ;LOAD SIO A CTL ADDRESS
```

```
2019    ED B3                  OTIR                     ;LOAD COMMAND TO SIO A
                       ;       INTIALIZE SIO CHANNEL B FOR INTERRUPT
201B    06 05                  LD      B,SIOBCL-SIOBCC ;LOAD COMMAND LENGTH
201D    0E 07                  LD      C,SIOBC         ;LOAD SIO B CTL ADDRESS
                       ;       NOTE THAT HL POINTS SIOBCC HERE
201F    ED B3                  OTIR                     ;LOAD COMMAND TO SIO B
                       ;
                       ;       INITIALIZE CONTROL FLAG
                       ;       B=CONTROL FLAG
                       ;       BIT  0  ADDRESS RECEIVED
                       ;       BIT  1  ONE CYCLE END
                       ;       BIT  2  CRC EXPECTED
                       ;       BIT  3  CRC 1 RE CEIVED
2021    06 00                  LD      B,0              ;RESET CONTROL FLAG
                       ;       INITIALIZE DATA
                       ;       D=DTA TO BE TRANSMITTED
                       ;       E=DATA TO BE EXPECTED
2023    11 0000                LD      DE,0             ;D=00H E=00H
2026    21 2400                LD      HL,BUFFER        ;SET BUFFER POINTER
                       ;       TEST BEGINS
2029    3E 55                  LD      A,ADRS           ;LOAD ADDRESS AS
                                                        ;THE FIRST DATA
202B    D3 04                  OUT     (SIOAD),A        ;TRANSMIT IT
202D    3E C0                  LD      A,0C0H           ;RESET TX UNDERRUN/EOM
202F    D3 05                  OUT     (SIOAC),A        ;
2031    FB                     EI                       ;ENABLE INTERRUPT
2032    18 FE                  JR      $                ;LOOP ETERNALLY
                       ;
                       ;       RE-ENABLE INTERRUPT
2034    FB             IEXIT:  EI                       ;RE-ENABLE INTERRUPT
2035    ED 4D                  RETI                     ;
                       ;
                       ;       TRANSMITTER BUFFER EMPTY INTERRUPT
2037    CB 48          I04:    BIT     1,B              ;ONE CYCLE END ?
2039    C2 2034                JP      NZ,IEXIT         ;YES,IEXIT
203C    7A                     LD      A,D              ;LOAD DATA
203D    D3 04                  OUT     (SIOAD),A        ;TRANSMIT DATA
203F    14                     INC     D                ;UPDATE DATA
2040    C2 2034                JP      NZ,IEXIT         ;ONE CYCLE END ?
                                                        ;NO,IEXIT
2043    CB C8                  SET     1,B              ;SET ONE CYCLE END
2045    C3 2034                JP      IEXIT            ;
                       ;
                       ;       RECEIVE CHARACTER AVAILABLE INTERRUPT
2048    DB 04          I06:    IN      A,(SIOAD)        ;READ DATA FROM SIO
204A    77                     LD      (HL),A           ;STORE IT INTO BUFFER
204B    23                     INC     HL               ;BUMP BUFFER POINTER
204C    CB 40                  BIT     0,B              ;ADDRESS RECEIVED ?
204E    CA 206D                JP      Z,I061           ;NO,I061
2051    CB 50                  BIT     2,B              ;RECEIVING CRC ?
2053    C2 2063                JP      NZ,I060          ;YES,060
2056    BB                     CP      E                ;DATA OK ?
2057    C2 208F                JP      NZ,DERROR        ;NO,DERROR
205A    1C                     INC     E                ;UPDATE DATA
205B    C2 2034                JP      NZ,IEXIT         ;ONE CYCLE END ?
                                                        ;NO,IEXIT
205E    CB D0                  SET     2,B              ;SET CRC RECEIVING
2060    C3 2034                JP      IEXIT
                       ;       RECEIVE CRC
2063    CB 58          I060:   BIT     3,B              ;CRC 1 RECEIVED ?
2065    C2 2034                JP      NZ,IEXIT         ;YES,IEXIT
2068    CB D8                  SET     3,B              ;SET CRC 2 EXPECTED
206A    C3 2034                JP      IEXIT            ;
```

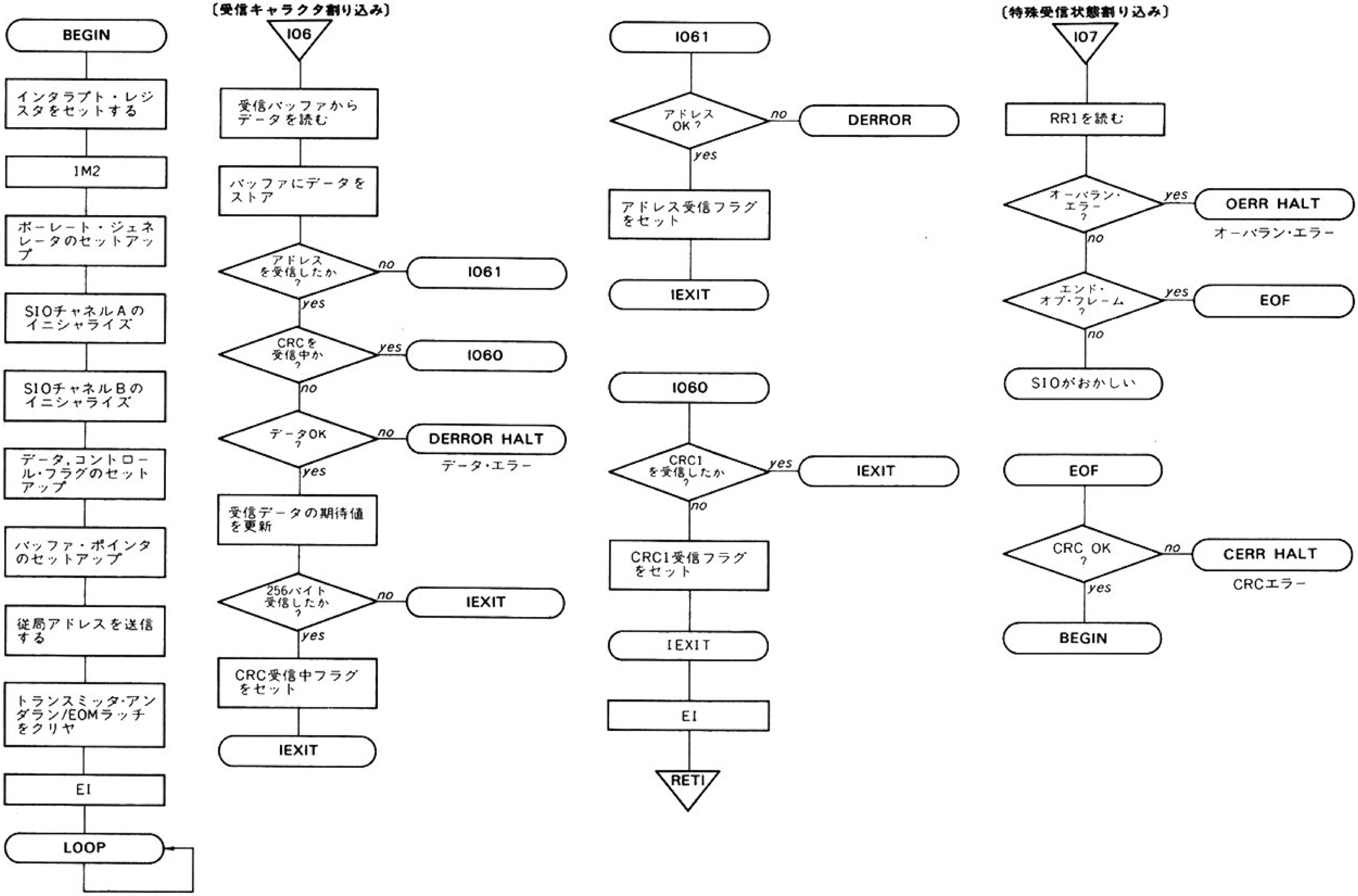
```
                           ;        ADDRESS RECEIVED
206D    FE 55              I061:    CP      ADRS                ;ADDRESS GOOD ?
206F    C2 208F                     JP      NZ,DERROR           ;NO,DERROR
2072    CB C0                       SET     0,B                 ;SET ADDRESS RECEIVED
2074    C3 2034                     JP      IEXIT               ;
                           ;
                           ;        SPECIAL RECEIVE CONDITION INTERRUPT
2077    3E 01              I07:     LD      A,01H               ;TO ACCESS RR1
2079    D3 05                       OUT     (SIOAC),A           ;
207B    DB 05                       IN      A,(SIOAC)           ;READ RR1
207D    CB 6F                       BIT     5,A                 ;RECEIVER OVERRRUN ?
207F    C2 2088                     JP      NZ,OERR             ;YES,OERR
2082    CB 7F                       BIT     7,A                 ;END OF FRAME ?
2084    C2 2089                     JP      NZ,EOF              ;YES,EOF
                           ;        SIO ERROR
2087    76                          HALT                        ;SOMETHING UNEXPECTED
                                                                ;HAPPENED
                           ;        OVERRUN ERROR
2088    76                 OERR:    HALT                        ;
                           ;
                           ;        END OF FRAME, NOW CHECK CRC RESULT
2089    CB 77              EOF:     BIT     6,A                 ;CRC GOOD ?
208B    CA 2000                     JP      Z,BEGIN             ;YES,BEGIN
                           ;        CRC ERROR
208E    76                 CERR:    HALT                        ;
                           ;        DATA ERROR
208F    76                 DERROR:  HALT                        ;DATA ERROR
                           ;
                           ;        SIO CHANNEL A COMMAND CHAIN
2090    98                 SIOACC:  DB      98H       ;WR0      RESET CHANNEL
                                                      ;         RESET TX CRC GENERATOR
2091    01                          DB      01H       ;WR0      POINTS WR1
2092    12                          DB      12H       ;WR1      INT ON ALL RX CHRACTERS
                                                      ;         TX INT ENABLE
2093    03                          DB      03H       ;WR0      POINTS WR3
2094    DD                          DB      0DDH      ;WR3      RX 8 BITS/CHARACTER
                                                      ;         ENTER HUNT PHASE
                                                      ;         RX CRC ENABLE
                                                      ;         ADDRESS SEARCH MODE
                                                      ;         RX ENABLE
2095    04                          DB      04H       ;WR0      POINTS WR4
2096    20                          DB      20H       ;WR4      X1 CLOCK
                                                      ;         SDLC MODE
2097    06                          DB      06H       ;WR0      POINTS WR6
2098    55                          DB      ADRS      ;WR6      SECONDARY STATION ADRS
2099    07                          DB      07H       ;WR0      POINTS WR7
209A    7E                          DB      7EH       ;WR7      FLAG
209B    05                          DB      05H       ;WR0      POINTS WR5
209C    EB                          DB      0EBH      ;WR5      DTR ON
                                                      ;         TX 8 BITS/CHARACTER
                                                      ;         TX ENABLE
                                                      ;         SDLC POLINOMIAL
                                                      ;         RTS ON
                                                      ;         TX CRC ENABLE
                           ;
209D                       SIOACL   EQU     $                   ;END OF SIOACC
                           ;
                           ;        SIO CHANNEL B COMMAND CHAIN FOR INTERRUPT
209D    18                 SIOBCC:  DB      18H       ;WR0      RESET CHANNEL
209E    01                          DB      01H       ;WR0      POINTS WR1
209F    04                          DB      04H       ;WR1      STATUS AFFECTS VECTOR
20A0    02                          DB      02H       ;WR0      POINTS  WR2
20A1    F0                          DB      0F0H      ;WR2      INTERRUPT VECTOR
                           ;
20A2                       SIOBCL   EQU     $                   ;END OF SIOBCC
                           ;
```

```
                            ;
20A2                                DS      64                  ;STACK AREA
20E2                        STACK   EQU     $                   ;STACK FRAME
                            ;
                            ;
                                    ORG     21F0H
                            ;       INTERRUPT VECTOR
21F0    0000                I00:    DW      0                   ;CH B TX BUFFER EMPTY
21F2    0000                        DW      0                   ;CH B EXTERNAL INT.
21F4    0000                        DW      0                   ;CH B RX CHAR. AVAIL.
21F6    0000                        DW      0                   ;CH B SPECIAL REC. COND.
21F8    2037                        DW      I04                 ;CH A TX BUFFER EMPTY
21FA    0000                        DW      0                   ;CH A EXTERNAL INT.
21FC    2048                        DW      I06                 ;CH A RX CHAR. AVAIL.
21FE    2077                        DW      I07                 ;CH A SPECIAL REC. COND.
                            ;
                            ;
                                    END
```

〈図49〉
Prog❺ のイニシャライゼーション・コマンド・チェイン(SDLCモード，割り込み駆動型)

	レジスタ	コマンド（16進数）	コマンドのビット・パターン 7 6 5 4 3 2 1 0	説明
チャネルA	WR0	98	1 0 0 1 1 0 0 0	リセット・チャネル CRCジェネレータをリセット
	WR0	01	0 0 0 0 0 0 0 1	WR1をポイント
	WR1	12	0 0 0 1 0 0 1 0	受信キャラクタ割り込み可 送信割り込み可
	WR0	03	0 0 0 0 0 0 1 1	WR3をポイント
	WR3	DD	1 1 0 1 1 1 0 1	受信8ビット/キャラクタ エンター・ハント・モード 受信CRCイネーブル アドレス・サーチ・モード レシーバ・イネーブル
	WR0	04	0 0 0 0 0 1 0 0	WR4をポイント
	WR4	20	0 0 1 0 0 0 0 0	クロックの倍率＝1 SDLCモード
	WR0	06	0 0 0 0 0 1 1 0	WR6をポイント
	WR6	55	0 1 0 1 0 1 0 1	従局アドレス
	WR0	07	0 0 0 0 0 1 1 1	WR7をポイント
	WR7	7E	0 1 1 1 1 1 1 0	フラグ・パターン
	WR0	05	0 0 0 0 0 1 0 1	WR5をポイント
	WR5	EB	1 1 1 0 1 0 1 1	DTR　ON 送信8ビット/キャラクタ トランスミッタ・イネーブル SDLC多項式 RTS　ON CRCジェネレータ・イネーブル
チャネルB	WR0	18	0 0 0 1 1 0 0 0	リセット・チャネル
	WR0	01	0 0 0 0 0 0 0 1	WR1をポイント
	WR1	04	0 0 0 0 0 1 0 0	ステータス・アフェクツ・ベクトル
	WR0	02	0 0 0 0 0 0 1 0	WR2をポイント
	WR2	××	× × × × × × × ×	割り込みベクトル

〈図50〉 Prog❺のフローチャート（SDLCモード，割り込み駆動型）



す前に，レジスタ HL にバッファ・アドレス，レジスタBにデータ・レングス（B＝0→256），レジスタCに SIO チャネルAのデータ・アドレスをセットします．

OTIR 命令により $\overline{\text{IORQ}}$ および $\overline{\text{WR}}$ が出されたときに，SIOのチャネルAがバッファ・エンプティでないときには $\overline{\text{W/RDYA}}$ は“L”になり，CPUにウェイトをかけます．バッファ・エンプティになると $\overline{\text{W/RDYA}}$ は“H”になり，1バイトのデータが SIO に対して送られます．カウントが256ですので以上の過程を256回繰り返します．

SIOのレディ機能

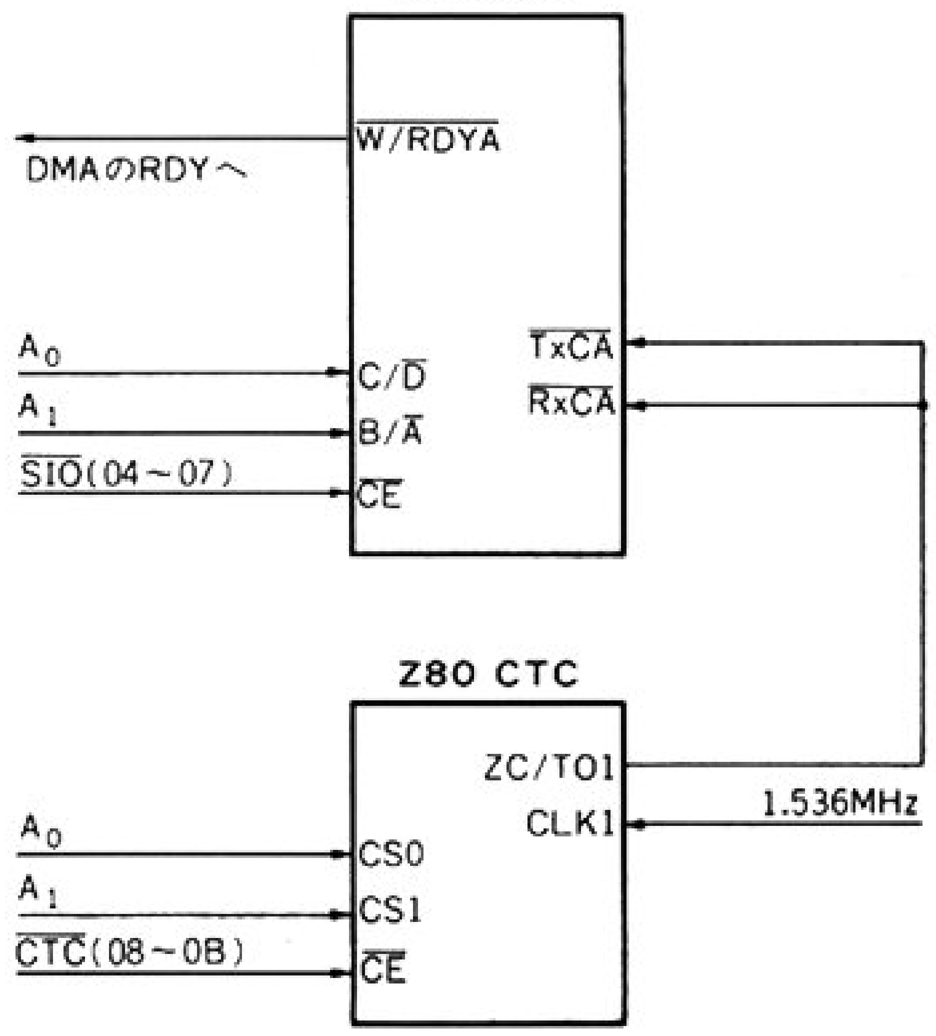
SIO には $\overline{\text{W/RDYA(B)}}$ ピンを DMA に対するレディ信号として使う機能があります．この機能は WR1 のビット7～5により選択することができます．Prog ❼に SIO のレディ機能を用い，Z80 DMA によりメモリ・データを SIO に送るプログラムを示します．また図52にこのプログラムが動作するための構成を示します．

このプログラムにおける SIO のイニシャライゼーションは Prog ❻ のそれとほとんど同じであり，異なるのは WR1 に対するコマンドだけです．DMAのプログラミングは，メモリ → I/O（SIO チャネルAデータ）とし，レディの有効レベルは“L”とします．

このプログラムはボーレート・ジェネレータ，および SIO，それに DMA をイニシャライズした後に，イネーブル DMA コマンドを出し動作を開始させます．

SIO はバッファ・エンプティのときには $\overline{\text{W/RDYA}}$ を“L”にします．これにより DMA は1バイトのデータをメモリから読み出し，それを SIO チャネルAデータに書き込みます．バッファ・エンプティでなくなると SIO は，$\overline{\text{W/RDYA}}$ を“H”にしますので，DMA

〈図52〉 Prog ❼ が動作する構成



のレディ・ラインは非有効レベルとなり，DMA はそれが有効レベルになるのを待ちます．

1ブロック（256 バイト）のデータ転送の終了は，DMAのステータス・レジスタに対するポーリングにより行います．DMA のステータス・バイトのビット5（エンド・オブ・ブロック）が“0”になると1ブロックのデータ転送が終了したことがわかります．

ここで注意すべきことは，DMA に対して何らかのコマンド（ここではステータス・リード）を出すと，DMA はその動作（バス・マスタであること）を中止してしまいます．

したがってステータス・リードを行うごとに，イネーブル DMA コマンドを出して DMA を再びイネーブルしてやる必要があります．

〈Prog ❻〉 SIO ウェイト機能

```
                  ;       Z80 SIO WAIT FUNCTION TEST
                  ;
                  ;       THIS PROGRAM TRANSMITS DATA TO SIO CHANNEL A
                  ;       USING WAIT FUNCTION.
                  ;
0009              CTC1    EQU     09H           ;CTC CHANNEL 1
                  ;
0004              SIOAD   EQU     04H           ;SIO CHANNEL A DATA
0005              SIOAC   EQU     SIOAD+01H     ;SIO CHANNEL A CONTROL
0006              SIOBD   EQU     SIOAC+01H     ;SIO CHANNEL B DATA
0007              SIOBC   EQU     SIOBD+01H     ;SIO CHANNEL B CONTROL
                  ;
                  ;
2400              BUFFER  EQU     2400H         ;DATA BUFFER
```

```
                      ;
0000                          ASEG
                              ORG     2000H
2000    F3            BEGIN:  DI                          ;DISABLE INTERRUPT
2001    31 2070               LD      SP,STACK            ;SET STACK POINTER
                      ;       SET UP BAUD RATE GENERATOR
2004    3E 47                 LD      A,47H               ;MODE WORD FOR CTC
2006    D3 09                 OUT     (CTC1),A            ;LOAD MODE WORD TO CTC
2008    3E 0A                 LD      A,10                ;SET TIME CONSTANT
200A    D3 09                 OUT     (CTC1),A            ;LOAD TIME CONSTANT
                      ;       INITIALIZE SIO CHANNEL A
200C    21 2029               LD      HL,SIOACC           ;LOAD COMMAND ADDRESS
200F    06 07                 LD      B,SIOACL-SIOACC     ;LOAD COMMAND LENGTH
2011    0E 05                 LD      C,SIOAC             ;LOAD SIO A CTL ADRS
2013    ED B3                 OTIR                        ;LOAD COMMAND TO SIO
                      ;       SET UP DATA AND SIO A DATA PORT ADDRESS
                      ;       B=DATA LENGTH
                      ;       HL=DATA ADDRESS
                      ;       C=SIO CHANNEL A DATA ADDRESS
2015    21 2400               LD      HL,BUFFER           ;LOAD BUFFER ADDRESS
2018    06 00                 LD      B,0                 ;SET COUNT=256
201A    75            BEGINO: LD      (HL),L              ;SET DATA INTO BUFFER
201B    23                    INC     HL                  ;BUMP BUFFER POINTER
201C    10 FC                 DJNZ    BEGINO              ;SET ALL ? NO,BEGINO
201E    21 2400       LOOP:   LD      HL,BUFFER           ;SET BUFFER ADDRESS
2021    06 00                 LD      B,0                 ;SET COUNT=256
2023    0E 04                 LD      C,SIOAD             ;LOAD DATA ADDRESS
                      ;       TRANSMIT DATA
2025    ED B3         XMIT:   OTIR                        ;TRANSMIT 256 BYTE OF
                                                          ;DATA USING WAIT.
2027    18 F5                 JR      LOOP                ;
                      ;
                      ;
                      ;       SIO CHANNEL A COMMAND CHAIN
2029    18            SIOACC: DB      18H       ;WR0      CHANNEL RESET
202A    01                    DB      01H       ;WR0      POINTS WR1
202B    80                    DB      80H       ;WR1      DISABLE ALL INTERRUPTS
                                                ;         WAIT ON TRANSMIT
202C    04                    DB      04H       ;WR0      POINTS WR4
202D    47                    DB      47H       ;WR4      X16 CLOCK
                                                ;         1 STOP BIT
                                                ;         EVEN PARITY
                                                ;         ENABLE PARITY
202E    05                    DB      05H       ;WR0      POINTS WR5
202F    EA                    DB      0EAH      ;WR5      DTR ON
                                                ;         TX 8 BITS/CHARACTER
                                                ;         TX ENABLE
                                                ;         RTS ON
                      ;
2030                  SIOACL  EQU     $                   ;END OF SIOACC
                      ;
2030                          DS      64                  ;STACK AREA
2070                  STACK   EQU     $                   ;
                      ;
                      ;
                              END
```

<Prog ❼> SIO レディ機能

```
                        ;       Z80 SIO READY FUNCTION TEST
                        ;
                        ;       THIS PROGRAM TRANSMITS DATA TO SIO CHANNEL A
                        ;       USING READY FUNCTION VIA Z80 DMA.
                        ;
0009                    CTC1    EQU     09H             ;CTC CHANNEL 1
                        ;
0004                    SIOAD   EQU     04H             ;SIO CHANNEL A DATA
0005                    SIOAC   EQU     SIOAD+01H       ;SIO CHANNEL A CONTROL
0006                    SIOBD   EQU     SIOAC+01H       ;SIO CHANNEL B DATA
0007                    SIOBC   EQU     SIOBD+01H       ;SIO CHANNEL B CONTROL
                        ;
0018                    DMA     EQU     18H             ;Z80 DMA ADDRESS
                        ;
2400                    BUFFER  EQU     2400H           ;DATA BUFFER
                        ;
0000                            ASEG
                                ORG     2000H
2000    F3              BEGIN:  DI                      ;DISABLE INTERRUPT
2001    31 20A3                 LD      SP,STACK        ;SET STACK POINTER
                        ;       SET UP BAUD RATE GENERATOR
2004    3E 47                   LD      A,47H           ;MODE WORD FOR CTC
2006    D3 09                   OUT     (CTC1),A        ;LOAD MODE WORD TO CTC
2008    3E 0A                   LD      A,10            ;SET TIME CONSTANT
200A    D3 09                   OUT     (CTC1),A        ;LOAD TIME CONSTANT
                        ;       INITIALIZE SIO CHANNEL A
200C    21 204B                 LD      HL,SIOACC       ;LOAD COMMAND ADDRESS
200F    06 07                   LD      B,SIOACL-SIOACC ;LOAD COMMAND LENGTH
2011    0E 05                   LD      C,SIOAC         ;LOAD SIO A CTL ADRS
2013    ED B3                   OTIR                    ;LOAD COMMAND TO SIO
                        ;       SET UP DATA AND SIO A DATA PORT ADDRESS
                        ;       B=DATA LENGTH
                        ;       HL=DATA ADDRESS
                        ;       C=SIO CHANNEL A DATA ADDRESS
2015    21 2400                 LD      HL,BUFFER       ;LOAD BUFFER ADDRESS
2018    06 00                   LD      B,0             ;SET COUNT=256
201A    75              BEGIN0: LD      (HL),L          ;SET DATA INTO BUFFER
201B    23                      INC     HL              ;BUMP BUFFER POINTER
201C    10 FC                   DJNZ    BEGIN0          ;SET ALL ? NO,BEGIN0
                        ;       INITIALIZE Z80 DMA
201E    21 2052                 LD      HL,DMACC        ;LOAD DMA COMMAND ADRS
2021    06 11                   LD      B,DMACL-DMACC   ;LOAD COMMAND LENGTH
2023    0E 18                   LD      C,DMA           ;LOAD DMA ADDRESS
2025    ED B3                   OTIR                    ;
                        ;       DMA IS INITIALIZED, NOW BEGIN TEST
2027    3E 01           LOOP:   LD      A,01H           ;PORT A --> PORT B
2029    D3 18                   OUT     (DMA),A         ;
202B    3E CF                   LD      A,0CFH          ;LOAD COMMAND
202D    D3 18                   OUT     (DMA),A         ;
202F    3E 05                   LD      A,05H           ;PORT B --> PORT A
2031    D3 18                   OUT     (DMA),A         ;
2033    3E CF                   LD      A,0CFH          ;LOAD COMMAND
2035    D3 18                   OUT     (DMA),A         ;
2037    3E 87                   LD      A,87H           ;ENABLE DMA
2039    D3 18                   OUT     (DMA),A         ;
                        ;       NOTE THAT DMA SHOULD BE RE-ENABLED AFTER
                        ;       READING OF STATUS BECAUSE DMA IS DISABLED
                        ;       BY READING OF STATUS.
203B    3E BF           LOOP0:  LD      A,0BFH          ;READ STATUS BYTE
203D    D3 18                   OUT     (DMA),A         ;
203F    DB 18                   IN      A,(DMA)         ;READ RR0
```

(次ページへつづく)

```
2041    CB 6F              BIT     5,A                ;END OF BLOCK ?
2043    3E 87              LD      A,87H              ;RE-ENABLE DMA
2045    D3 18              OUT     (DMA),A            ;
2047    20 F2              JR      NZ,LOOPO           ;GO TO LOOPO IF NOT
                                                      ;END OF BLOCK
2049    18 DC              JR      LOOP               ;TRY AGAIN TRANSFER
                  ;
                  ;
                  ;        SIO CHANNEL A COMMAND CHAIN
204B    18        SIOACC:  DB      18H       ;WRO     CHANNEL RESET
204C    01                 DB      01H       ;WRO     POINTS WR1
204D    CO                 DB      OCOH      ;WR1     DISABLE ALL INTERRUPTS
                                             ;        READY ON TRANSMIT
204E    04                 DB      04H       ;WRO     POINTS WR4
204F    47                 DB      47H       ;WR4     X16 CLOCK
                                             ;        1 STOP BIT
                                             ;        EVEN PARITY
                                             ;        ENABLE PARITY
2050    05                 DB      05H       ;WRO     POINTS WR5
2051    EA                 DB      OEAH      ;WR5     DTR ON
                                             ;        TX 8 BITS/CHARACTER
                                             ;        TX ENABLE
                                             ;        RTS ON
                  ;
2052              SIOACL   EQU     $                  ;END OF SIOACC
                  ;
                  ;
                  ;        Z80 DMA INITIALIZATION COMMAND CHAIN
2052    C3        DMACC:   DB      OC3H      ;WR6     RESET DMA
2053    C3                 DB      OC3H      ;
2054    C3                 DB      OC3H      ;
2055    C3                 DB      OC3H      ;
2056    C3                 DB      OC3H      ;
2057    C3                 DB      OC3H      ;
2058    7D                 DB      7DH       ;WRO     TRANSFER
                                             ;        PORT A --> PORT B
                                             ;        PORT A ADDRESS FOLLOWS
                                             ;        PORT A LENGTH FOLLOWS
2059    2400               DW      BUFFER    ;        PORT A ADDRESS
205B    0100               DW      256       ;        PORT B LENGTH
205D    14                 DB      14H       ;WR1     PORT A IS MEMORY
                                             ;        PORT A ADRS INCREMENTS
                                             ;        NO TIMING BYTE FOLLOWS
205E    28                 DB      28H       ;WR2     PORT B IS I/O
                                             ;        PORT B ADDRESS FIXED
                                             ;        NO TIMING BYTE FOLLOWS
205F    80                 DB      80H       ;WR3     DISABLE INTERRUPT
2060    85                 DB      85H       ;WR4     BYTE MODE
                                             ;        PORT B ADDRESS FOLLOWS
2061    04                 DB      SIOAD     ;        PORT B ADDRESS
2062    82                 DB      82H       ;WR5     STOP ON END OF BLOCK
                                             ;        CE/ ONLY
                                             ;        READY ACTIVE LOW
                  ;
2063              DMACL    EQU     $                  ;END OF DMACC
                  ;
2063                       DS      64                 ;STACK AREA
20A3              STACK    EQU     $                  ;
                  ;
                  ;
                           END
```

付　録

Appendix 1　Z80シングル・ボード・コンピュータとモニタ・プログラム

Z80 CPUおよびその周辺LSIの動作を理解するには，これらのICを使ったマイクロコンピュータを作ってみて，実際に動作させてみるのが一番手っ取り早い方法です．

しかし，マイクロコンピュータに限らず，コンピュータというものは，プログラムを実行させることにより機能を果たすように設計されていますので，テスト・プログラムのようなものがなければ，動作しているのかしていないのか皆目見当がつきません．

昔はマイクロコンピュータを作る場合には，必ずといっていいくらい，トグル・スイッチとLEDによる操作パネルを付け，このパネルからショート・プログラムをキー・インし，このプログラムにより，自分で作ったマイクロコンピュータが思いどおりに動くか否かを判別していました．

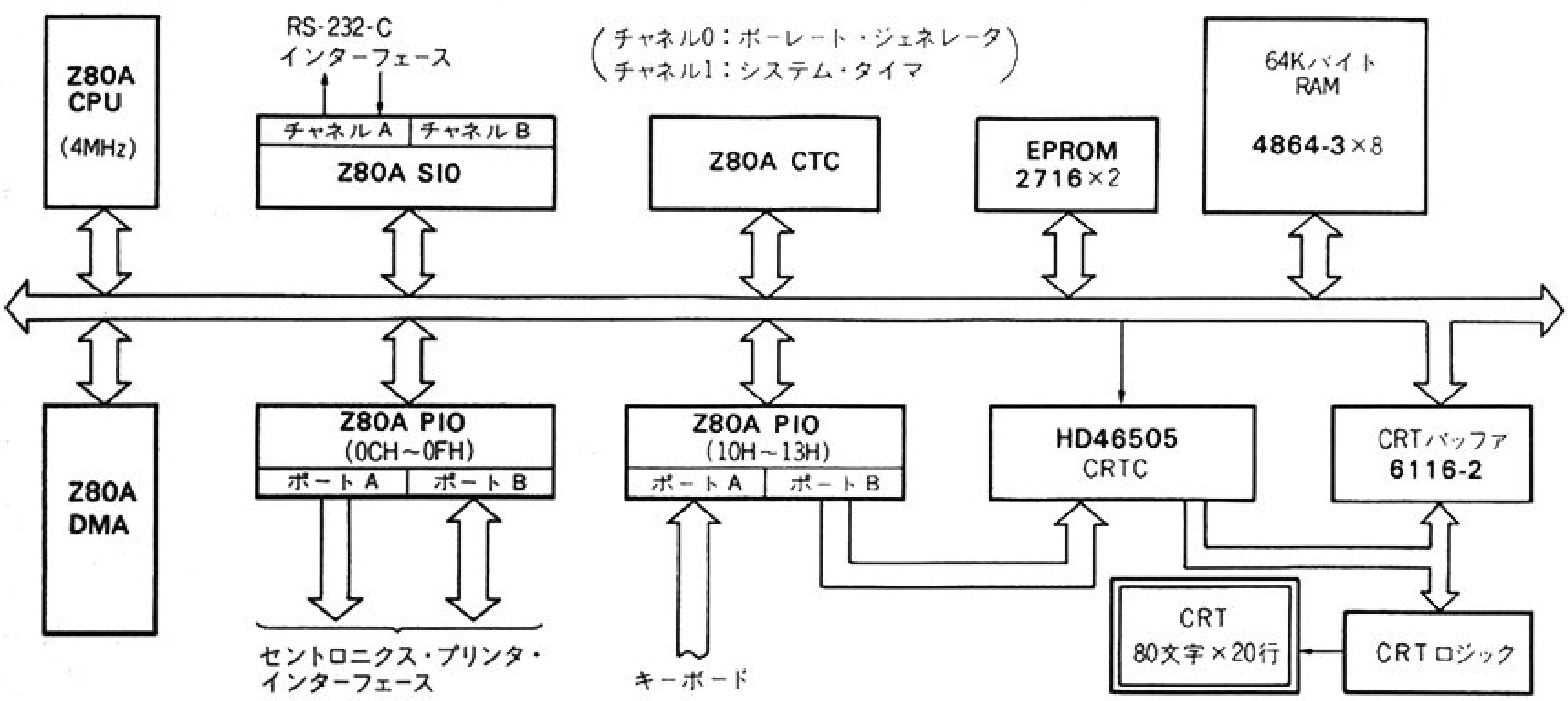
現在ではCPUや周辺LSIの動作に馴れ，これらに関してわからないことはさほどないので，スイッチやLEDを用いたパネルを作ることはせず，代わりにモニタ・プログラムを書き，これをEPROMに焼き付けることにより，スイッチやLEDによるパネルの機能を果たさせることも可能です．

ここで説明するモニタ・プログラムは，シングル・ボード・コンピュータのハードウェアを作ったときに，その動作を確認するために，作り上げたプログラムです．プログラムというものはハードウェアの動作が完全に近くないと動かないにもかかわらず，「ハードウェアの動作を確認するためのプログラムを作る」ということがらには明らかな自己矛盾がありますが，絶対に動くという自信を持ったハードウェアを作れた場合には，このようなことも可能です．これにはかなり「運」もありますが….

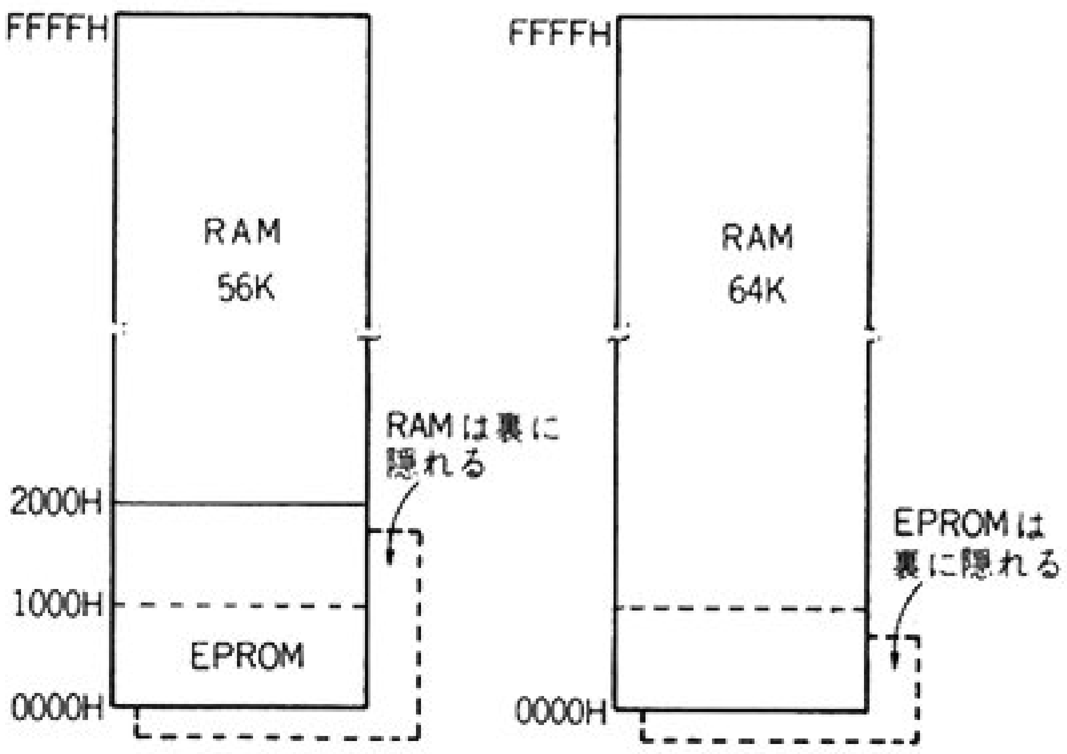
ここで示すモニタ・プログラムは，やはりここで示すハードウェア上でのみしか動作しませんが，Z80の周辺LSIがすべて使われていますので，Z80関係のプログラムを勉強する際のよい例になると思います．

また，ハードウェアに関しても，Z80 CPUと64 K DRAMやCRTコントローラ，HD46505(6845)とのインターフェースなどは，誰にとっても興味のあるものであろうと思います．

〈図1〉シングル・ボード・コンピュータの構成



〈図2〉 メモリ・マップ



(a) リセット時　(b) ディスエーブルROMを行ったとき

シングル・ボード・コンピュータのハードウェア

図1に製作したシングル・ボード・コンピュータのハードウェア構成を示します．CPUはZ80A CPUであり，クロックは4MHzです．以下に各構成部について説明します．

■メモリ(EPROM，RAM)

メモリは4KバイトのEPROM(2716×2)と64KバイトのDRAMです．図2にメモリ・マップを示します．リセット時には，始めの8Kバイト(0000H～1FFFH)はEPROMの空間となり，後の56KバイトがRAMです．この場合，始めの8Kバイトに対するリード命令はEPROMに対して行われ，ライト命令は裏に隠れたRAMに対して行われます．

I/O命令により，“ディスエーブルROM”を行うと，64Kバイトの全空間がRAMとなります．この場合にはEPROMは完全に裏に隠れてしまい，始めの8Kバイトに対するリード/ライト命令は，ともにRAMに対して行われます．

■Z80周辺LSI，およびI/Oポート

使用したZ80周辺LSIは，DMA，SIO，CTC，ならびに2個のPIOです．DMAはメイン・メモリとCRTリフレッシュ・バッファ間のDMAデータ転送専用に用いています．

SIOはRS-232-Cインターフェースのシリアル・コミュニケーション用で，CTCはシステム・クロックとSIOのボーレート・ジェネレータとして用いています．

2個のPIOの中の一つは，セントロニクス・プリンタ・インターフェースとして用い，他の一つはキーボードとCRTコントローラ，HD 46505とのインターフェースに用いています．

このほかに，ふつうのI/Oポートとして，DIPスイッチを読み取る入力ポートがあり，出力ポートとしては，

・LEDを点灯させるポート
・NMIフリップフロップをクリヤするポート
・ディスエーブルROMを行うポート
・HD46505のレジスタ・セレクトを行うポート
・DMAのRDYをクリヤするためのポート

があります．図3にI/Oポートの一覧表を示します．

■CRTコントロール・ロジック

図4にCRTディスプレイの表示画面構成を示します．ドット・クロックは16MHzで，1文字は8×10ドットの文字セル上に7×9ドットで表示されます．1画面は80文字(横)×20行(縦)で構成され，合計1600文字が画面に表示されます．

当初は80文字×25行の表示でいく予定だったのですが，使用するCRTモニタが古い家庭用白黒テレビであるためと，手持ちのキャラクタ・ジェネレータが7×9ドットのものであったため，このような画面構成になってしまいました．

〈図3〉 I/Oポートの一覧表

00～03	R/W	SIO	00	チャンネルAデータ(RS-232-Cインターフェース)
			01	チャンネルAコントロール
			02	チャンネルBデータ
			03	チャンネルBコントロール
04	R/W	DMA		メイン・メモリとCRTバッファ間のDMA転送用
08～0B	R/W	CTC	08	チャンネル0(ボーレート・ジェネレータ)
			09	チャンネル1(システム・クロック)
			0A	チャンネル2
			0B	チャンネル3
0C～0F	R/W	PIO	0C	ポートAデータ(セントロニクス・インターフェースのデータ)
			0D	ポートAコントロール
			0E	ポートBデータ(セントロニクス・インターフェースのコントロール)
			0F	ポートBコントロール
10～13	R/W	PIO	10	ポートAデータ(キーボード)
			11	ポートAコントロール
			12	ポートBデータ(HD 46505のデータ・ライン)
			13	ポートBコントロール
18	R	DIPスイッチのリード		
	W	LEDへのライト		
19	W	NMIフリップフロップのクリヤ		
1A	W	ディスエーブルROM		
1B	W	HD 46505のレジスタ・セレクト(D_0=0：アドレス・レジスタ，D_0=1：コントロール・レジスタ)		
1C	W	DMARDYのクリヤ		

水平同期信号の周波数は17.7kHzであり，垂直同期信号のそれは65Hzです．白黒テレビとのインターフェースは，コンポジット・ビデオで，テレビの同期分離回路の出力に接続しています．前記の水平/垂直同期信号の周波数は，標準テレビジョンの周波数とは若干異なりますが，十分に同期のとれる範囲内にあります．

図5にCRTコントロール・ロジックの構成を示します．CRTCはHD46505(日立)で，PIOとCRTCインターフェース回路により，プログラミングを行っています．

CRTバッファ・メモリ(リフレッシュ・バッファ・メモリ)は，2Kバイトのスタティック RAMです．このCRTバッファ・メモリとメイン・メモリ(64Kバイトのメモリ)とのデータ転送は垂直ブランキング期間中に，DMAデータ転送により行われます．CRTバッファ・メモリはI/O空間に置かれており，かつDMAデータ転送時($\overline{BUSAK}$="L")にのみセレクトされるようになっています．

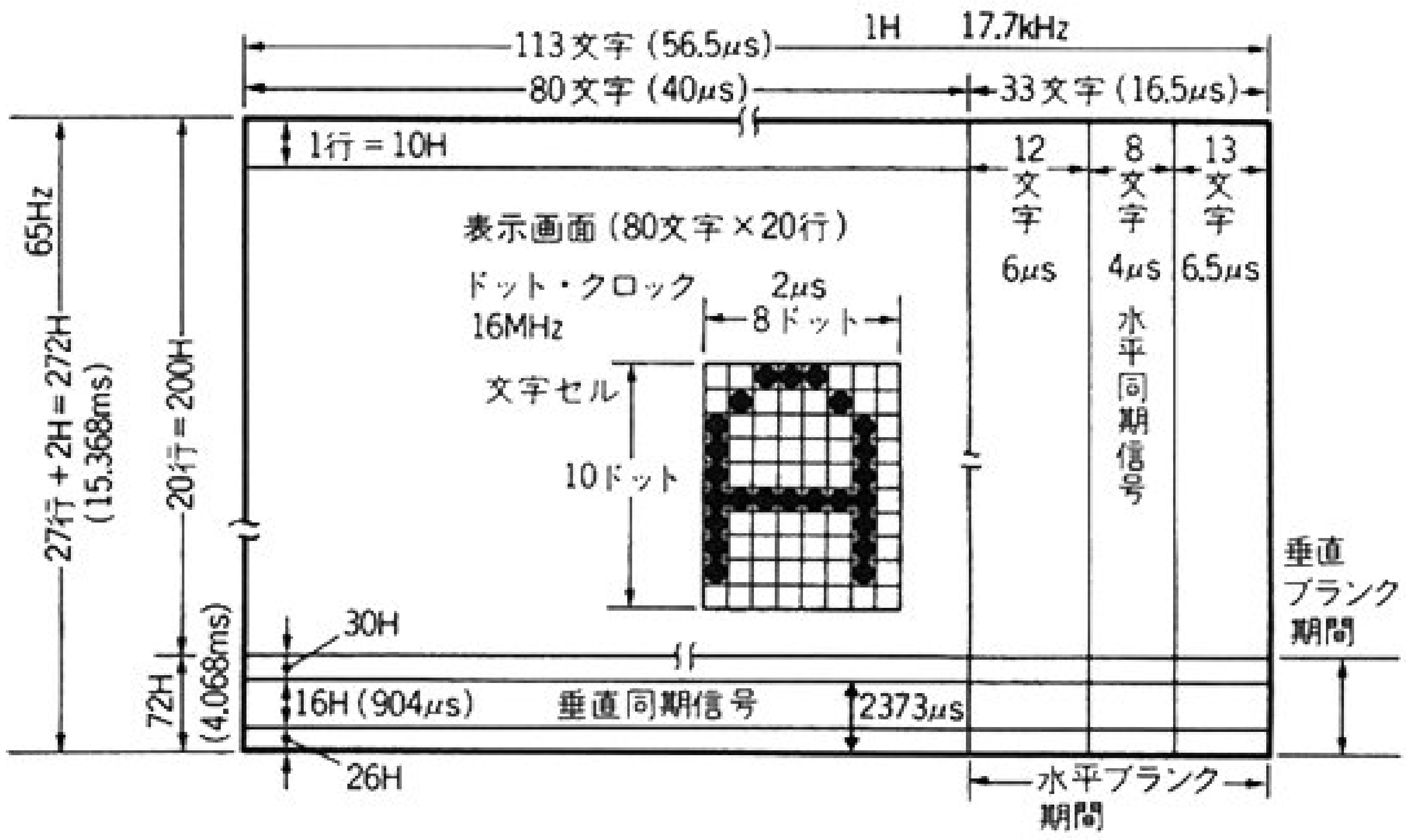
図6にメイン・メモリとCRTバッファ・メモリ間のDMAデータ転送のタイミングを示します．モニタのプログラムは，メイン・メモリ中のスクリーン・データ(1600バイト)をCRTバッファ・メモリにDMAデータ転送を行うように，Z80 DMAをコンティニュアス・モードでプログラミングします．

垂直同期信号(VS)が立ち上がると，DMAコントロール・ロジック内のワンショット(パルス幅=4.6ms)がトリガされ，DMAレディ信号($\overline{DMARDY}$)がアクティブになります．これにより，メイン・メモリ→CRTバッファ・メモリ間のDMAデータ転送が開始されます．このDMAデータ転送の方向は，メモリ→I/Oです．

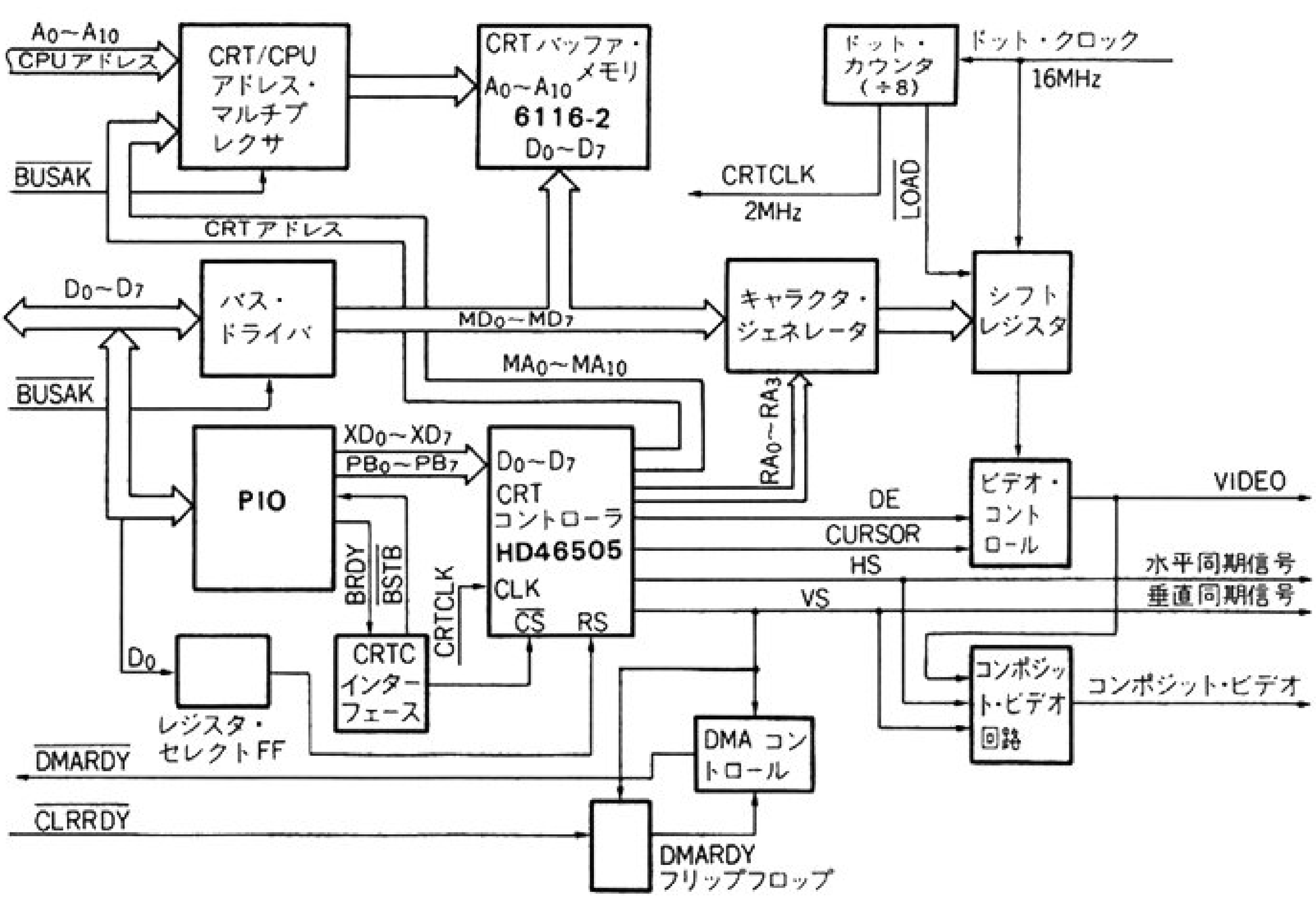
1600バイトのデータが転送し終ると，Z80 DMAはエンド・オブ・ブロックの割り込みを発生します．この割り込みを受けると，モニタのプログラムは$\overline{CLRRDY}$信号を出力して，$\overline{DMARDY}$信号を"H"にクリヤし，再度Z80 DMAをイネーブルします．

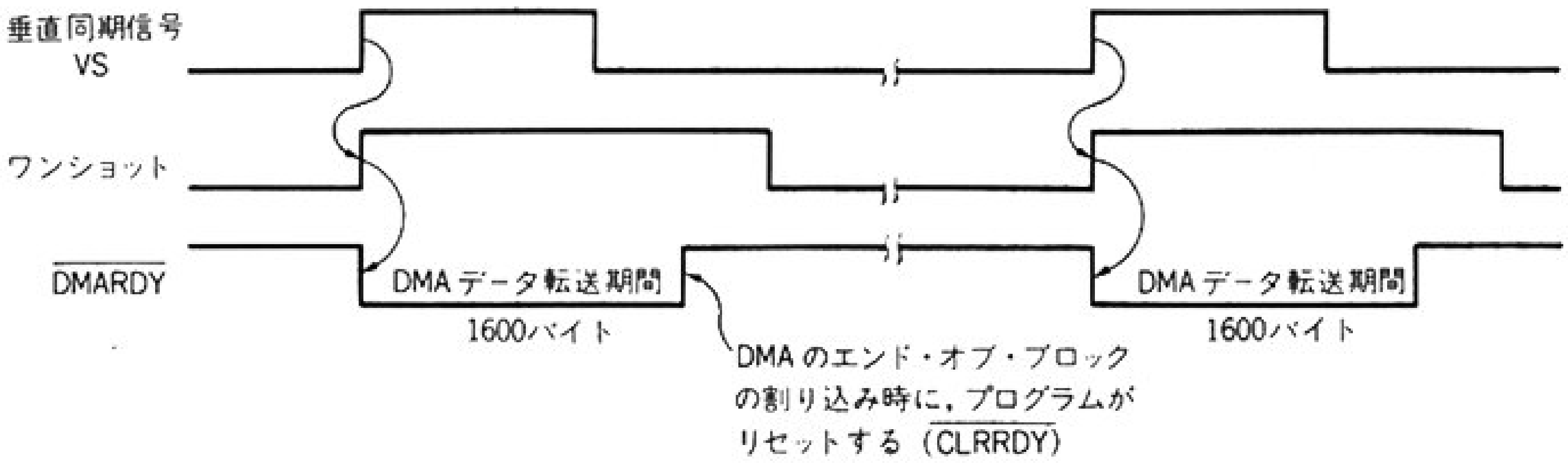
以上のように，メイン・メモリ→CRTバッファ・メモリのデータ転送は，垂直

〈図4〉表示画面の構成



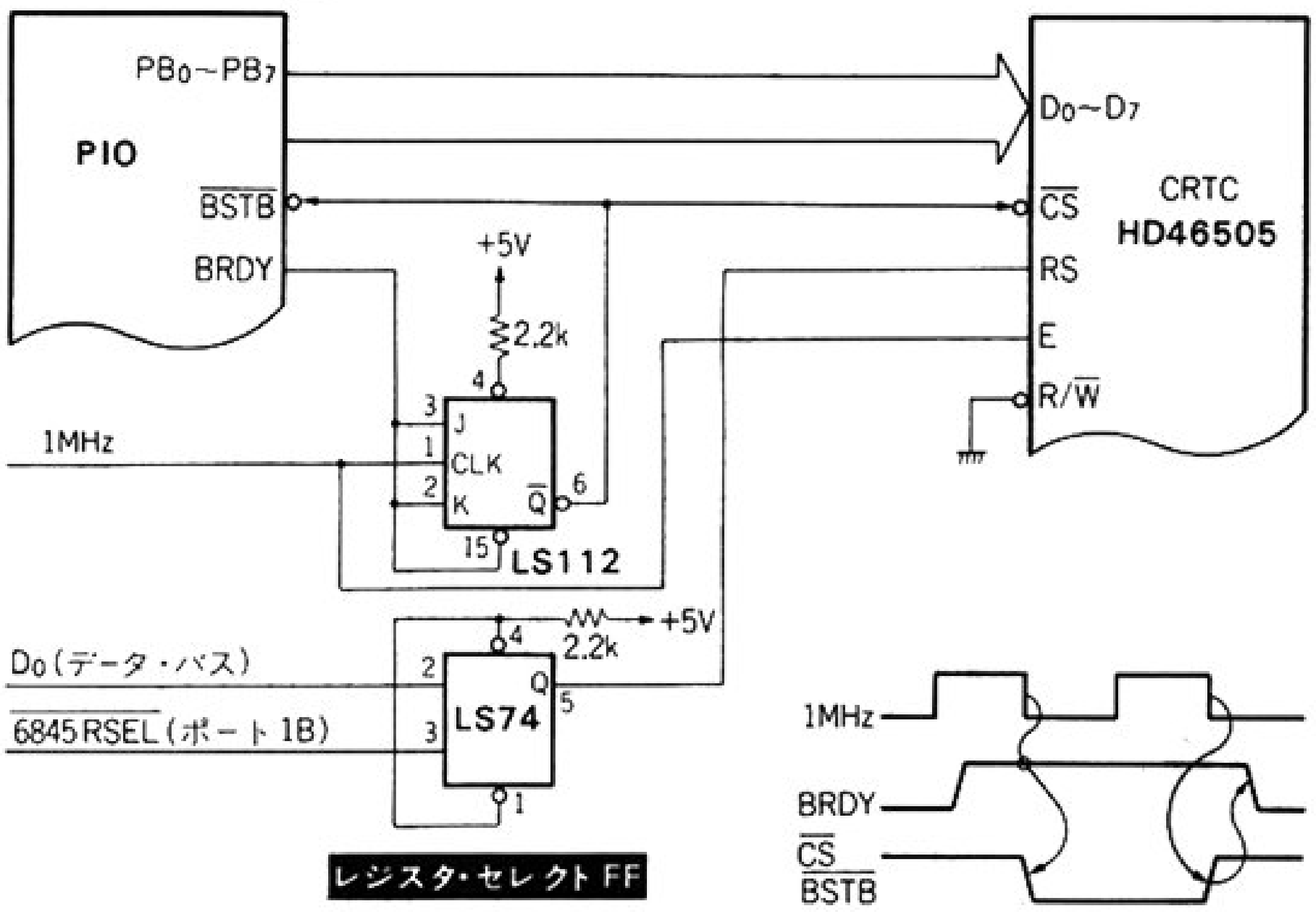
〈図5〉 CRTコントロール・ロジック構成





〈図6〉 CRTバッファ・メモリに対するDMAデータ転送

〈図7〉 CRTC HD 46505とのインターフェース



同期信号が立ち上がってからの垂直ブランキング時間内に行われますが，図4からこの時間は2373 μsとなります．

Z80 DMAのメモリ→I/Oデータ転送は，標準タイミングでは1バイトのデータを転送するのに，7クロック・サイクル(メモリ＝3, I/O＝4)を要します．これはシステム・クロックを4 MHzとした場合，1.75 μsに相当します．

したがって，Z80 DMAの標準タイミングを使用すると，1600バイトのデータを転送するのに，2800 μs(1.75 μs×1600)かかってしまいます．ところが，データ転送に使える時間は2373 μsなのです．この問題はZ80 DMAの可変タイミング・サイクル機能を用いて解決してあります．

つまり，可変タイミング・サイクル機能を用いて，I/Oサイクルを2サイクルにしたのです．こうすると，1バイトのデータを転送するのに必要な時間は，5サイクル(1.25 μs)となり，1600バイトのデータを転送するのにかかる時間は，2000 μs(1.25 μs×1600)となり，前記の2373 μs以下となります．

次にCRTC, HD46505とのインターフェースについて説明します．図7にCRTCとのインターフェース回路を示します．CRTCのデータ・バス(D_0～D_7)はPIOのPB_0～PB_7に接続されています．PIOのポートBはモード0(ストローブ付き出力)にプログラミングします．

CRTCに対してデータを書き込む(プログラミングする)ときには，まずデータのビット0(D_0)を0か1にし，ポート1Bに対してOUT命令を出し，レジスタ・セレクトFFを所望の状態にセットします．

このレジスタ・セレクトFFの出力は，CRTCのRSに接続されており，ポート1Bに対するOUT命令のデータのビット0が1ですと，レジスタ・セレクトの状態になります．

次にCRTCに対して書き込みたいデータを，PIOのポートBに出力します．PIOのポートBに対してデータが出力されると，PIOのBRDYが"H"になり，1 MHzのクロックの1周期分だけ，CRTCの$\overline{CS}$が"L"になります．

この$\overline{CS}$に送られている信号は，PIOの$\overline{BSTB}$としても用いられており，$\overline{BSTB}$の立ち上がりにより，BRDYは"L"に戻ります．PIOのポートB上のデータは$\overline{CS}$が立ち上がるころに，CRTCに書き込まれます．

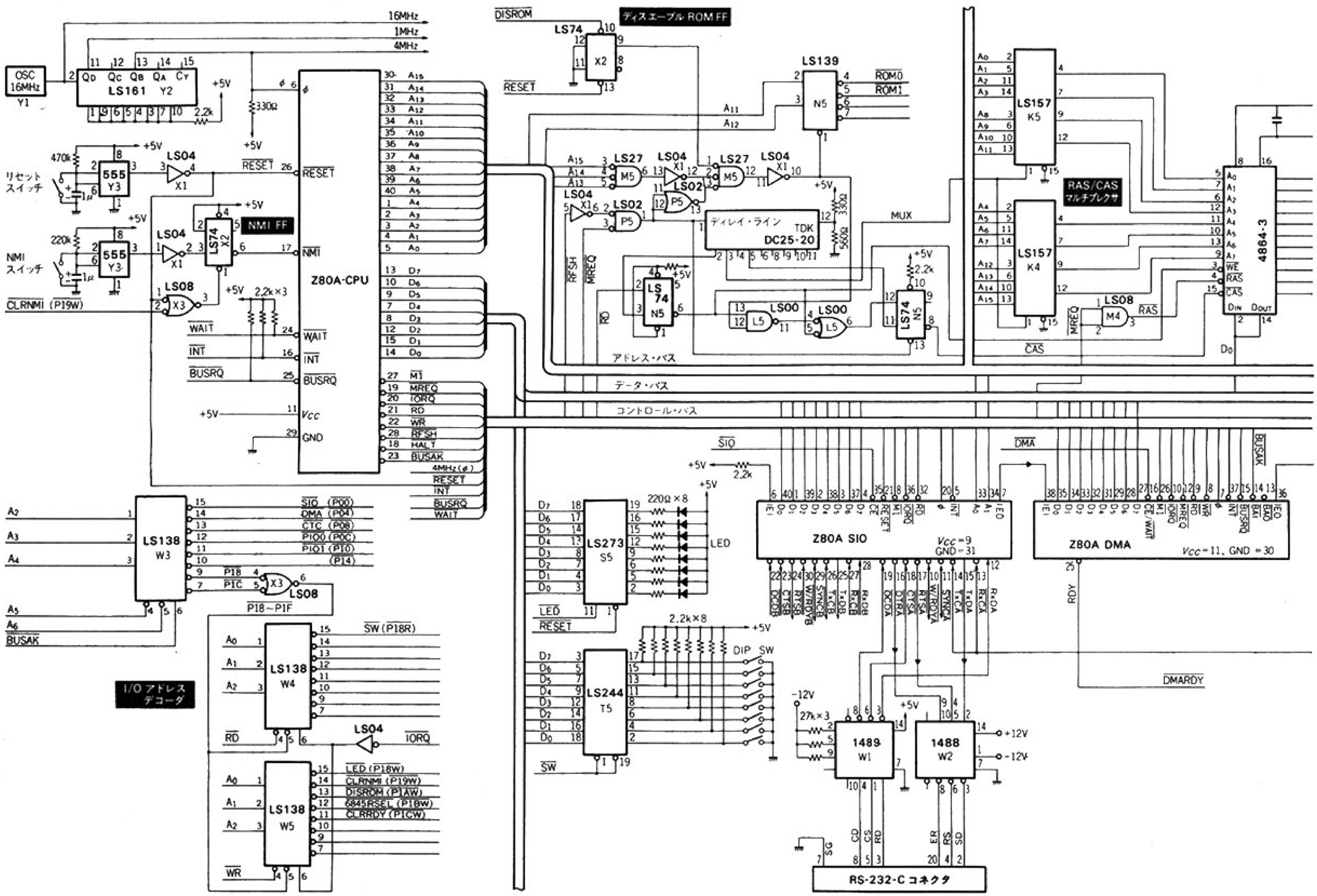
なお，このロジックはCRTCに対する書き込みだけしか考慮してありません．CRTCにはカーソル・レジスタやライト・ペン・レジスタがあり，これらのレジスタを読むこともできるのですが，本応用では使用していませんので，CRTCからのリード動作はロジックに含めませんでした．

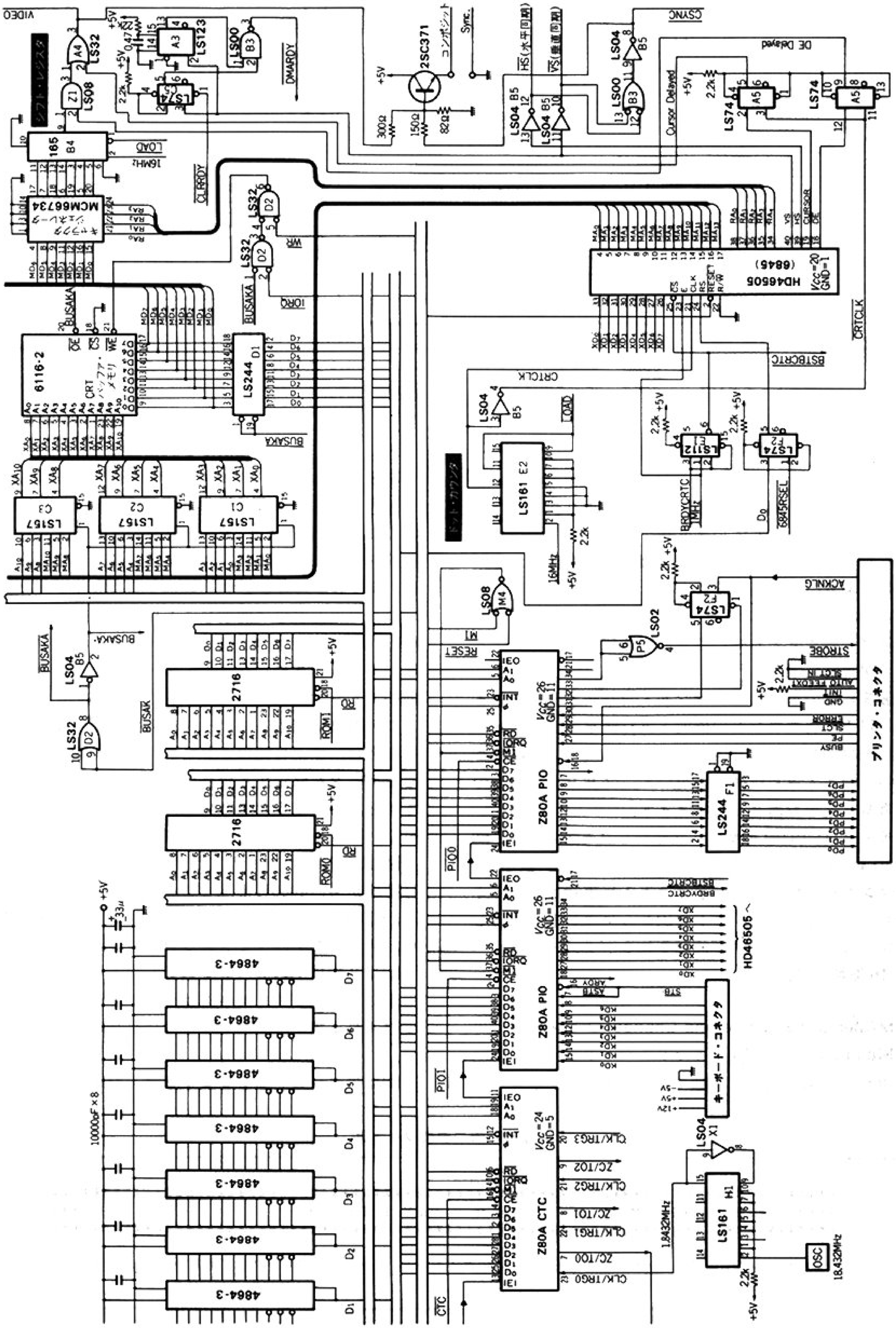
図8に製作したシングル・ボード・コンピュータの全回路図を示します．

モニタ・プログラム

以下に製作したZ80シングル・ボード・コンピュータ用のモニタ・プログラムの機能について説明をします．ここではモニタの機能と使いかたについてのみ説明をし，プログラムの内容そのものについては説明をしませんが，興味ある読者は本稿末に示すプログラム・リストを解読してみてください．

〈図8〉Z80シングル・ボード・コンピュータの全回路図





VIDEO
LS32
LS08
LS74
LS123
LS00
DMARDY
2SC371
コンポジット
Sync.
CSYNC
LS04
DE Delayed
Cursor Delayed
165
LOAD
16MHz
CLRRDY
MCM66734
6116-2
LS244
LS157
HD46505
(6845)
Vcc=20
GND=1
CRTCLK
BSTBCRTC
LS161
LS112
16MHz
6845RSEL
BUSAKA
BUSAK
2716
ROM0
ROM1
RD
IORQ
WR
RESET
M1
PIO0
PIO1
Z80A PIO
Z80A CTC
LS02
ACKNLG
STROBE
プリンタ・コネクタ
キーボード・コネクタ
HD46505へ
4864-3
10000pF×8
CLK/TRG0
ZC/TO0
CLK/TRG1
ZC/TO1
CLK/TRG2
ZC/TO2
CLK/TRG3
1.8432MHz
OSC
18.432MHz
CTC

〈図 9〉 モニタのコマンド・レパートリ

コマンド	コマンド名	コマンド・フォーマット	機能
M	Memory Examine and Change	Maaaa [CR]	メモリの内容を表示し，それに変更を加える
P	Port Examine and Change	Paa [CR]	ポートの内容を表示し，それに変更を加える
D	Display Memory	Daaaa, llll [CR]	メモリの内容を表示する
A	Add Binary Value	Aaaaa, bbbb [CR]	二つの2進数の和を求める
S	Subtract Binary Value	Saaaa, bbbb [CR]	二つの2進数の差を求める
F	Fill Memory with Word	Faaaa, llll, dddd [CR]	メモリにデータ(ワード)を書き込む
V	Move Memory	Vaaaa, bbbb, llll [CR]	メモリ上のデータを移動する
C	Compare Memory	Caaaa, bbbb, llll [CR]	メモリ上のデータを比較する
O	Set Offset	Oaaaa [CR]	アドレス・オフセットをセットする
Q	Inquire Offset	Q [CR]	オフセット値を表示する
L	Load Program	L [CR]	シリアル・ポートからプログラムをロードする
G	Go to Program	Gaaaa [CR]	プログラムにコントロールを移す

aaaa　メモリ・アドレス　　bbbb　メモリ・アドレス　　llll　レングス(長さ)
aa　I/Oポート・アドレス　　dddd　データ(2バイト)　　[CR]　キャリッジ・リターン

ここで示すモニタは，筆者の製作したシングル・ボード・コンピュータが，設計どおりに動くか否かを確かめる目的で数年前に作ったもので，あまり複雑な機能を含ませることはしていません．

したがって，このモニタの機能は，旧態依然としたものであり，数年前のそれとはほとんど違いはありません．しかしながら，このモニタがあれば，これをデバッギング・エイドとして，さらに高機能なモニタを作ることが可能です．

図9にこのモニタのコマンド・レパートリを示します．以下に各コマンドの機能と実行例を示しますが，実行例はCRTディスプレイの表示ではなく，このモニタのハード・コピー機能を用いて，プリントしたものであることをお断りしておきます．

モニタはCPUのリセット状態から立ち上がると，図10のようなタイトルをスクリーンの最下部に表示し，コマンド・プロンプト'>'をカラム1に表示します．オペレータはこのプロンプトの次からコマンドを入力します．なお，コマンドで使用されるパラメータはすべて16進数です．

・M(Memory Examine and Change)コマンド
>Maaaa [CR]　（[CR]：キャリッジ・リターン）

コマンドMに続いて，調べたいメモリのアドレスを入力します．モニタは指定されたメモリの内容とハイフンを表示します．メモリの内容を変更しないときには[CR]を押し，変更するときにはデータ(1バイト)を入力し，[CR]を押すと，モニタは次のアドレスの内容とハイフンを表示します．コマンドを終了させるときには，文字Xを入力します．

このコマンドでは便利なことに，バック・スペース(BS)キーが使えます．モニタがデータとハイフンを表示しているときに，バック・スペース・キーを押すと，前のアドレス(1だけ少ない)のデータを表示します．

この機能は間違ってデータを入力してしまったような場合に，すぐに元のアドレスに戻り，間違えて入力したデータを修正する場合に便利です．通常のモニタですと，このような場合には，いったんコマンドを終結し，再度コマンドを入力し直す必要があります．

図11にMコマンドの実行例を示します．(a)では4000H番地から，4009H番地までのデータを01H，23H，…，CDH，EFHと変更し，一度コマンドを終結した後に，入力したデータをチェックしています．

(b)は2桁以上のデータを入力したときはどうなるかを示したものです．モニタはデータとして何文字でも受け入れますが，有効となるのは最後の2文字だけです．

(c)はアドレスとして4文字を越えたものを入力した

〈図 10〉 モニタのタイトル

```
*************************************************************
****    Z80 SCREEN MONITOR    VER A    JULY 12TH 1983    ****
*************************************************************
>
```

モニタはCPUのリセットから立ち上がると，このタイトルと，コマンド・プロンプトである'>'を表示する．

場合のモニタの動作を示しています．この例では8文字(12345678)が入力されていますが，最後の4文字(5678)が有効になっています．

(d)はバック・スペース機能の使いかたを示しています．3000H番地にデータ56Hを入力した後に，間違いに気づき，バック・スペース・キーにより前の番地に戻し，正しいデータ45Hを入力しています．

このコマンドではメモリに書き込んだデータを読んで，正しく書き込まれたか否かのチェックをしています．

(e)はこのチェックの結果，エラーが検出された場合です．メモリの最初の4KバイトはEPROMですので，書き込むことはできません．そのため，*CAN NOT WRITEというエラー・メッセージが出されています．

・P(Port Examine and Change)コマンド

>Paa CR

コマンドPの後に2桁のI/Oポート・アドレスを入力します．モニタは指定されたポートのデータを読み取り，そのデータとハイフンを表示します．ポートにデータを書き込む場合には，ハイフンの後にデータと CR を入力します．

データを書き込む必要がない場合には，CR のみを押します．コマンドを終結させるには，文字Xを入力します．Mコマンドの場合と異なり，このPコマンドでは CR が入力されるごとに，I/Oポート・アドレスが上がっていくことはなく，常に指定されたアドレスのま

〈図11〉 Mコマンドの実行例

```
>M4000
*M4000 CD-01
*M4001 EF-23
*M4002 01-45
*M4003 63-67
*M4004 CD-89
*M4005 EF-AB
*M4006 01-CD
*M4007 63-EF
*M4008 CC-X
>M4000
*M4000 01-
*M4001 23-
*M4002 45-
*M4003 67-
*M4004 89-
*M4005 AB-
*M4006 CD-
*M4007 EF-
*M4008 CC-X
>
```

(a)

メモリにデータを書き込み，それをチェックする

```
>M2000
*M2000 DE-123456
*M2001 FF-X
>M2000
*M2000 56-X
>
```

(b)

データを3文字以上入力したときは，最後の2文字が有効

```
>M12345678
*M5678 CC-X
>
```

(c)

アドレスを5文字以上入力したときは，最後の4文字が有効

```
>M3000
*M3000 CC-56
*M3001 FF-
*M3000 56-45
*M3001 FF-X
>
```

(d)

3001H番地のデータを表示したときに，バックスペース・キーにより，3000H番地に戻す

```
>M0000
*M0000 31-22*CAN NOT WRITE
*M0001 00-33*CAN NOT WRITE
*M0002 00-X
>
```

(e) メモリにデータが書き込めないとき

〈図12〉 Pコマンドの実行例

```
>P18
*P18 A1-11
*P18 A1-22
*P18 A1-33
*P18 A1-44
*P18 A1-
*P18 A1-
*P18 A1-
*P18 A5-  } ここで DIP スイッチの
*P18 85-X } 状態を変化させた
```

〈図13〉 Dコマンドの実行例

```
>D200,100
*M0200 ED 4D DB 12 CB 47 20 F4 3E 00 D3 12 3E 40 D3 12   .M...G .>...>@..
*M0210 7E D3 10 3E C0 D3 12 3E 40 D3 12 3A B2 F7 CB F7   ...>...>@..:....
*M0220 32 B2 F7 23 01 00 FF CD 4F 02 20 03 21 B3 F7 22   2..#....O. .!.."
*M0230 AF F7 21 B2 F7 CB BE 18 C3 F5 C5 E5 21 B2 F7 CB   ..!.........!...
*M0240 76 28 B9 CB B6 3E 00 D3 12 3E 40 D3 12 18 9A 78   .(...>...>@.....
*M0250 BC C0 79 BD C9 F5 E5 D5 C5 DD E5 2A 44 F7 DD 2A   ...........*D..*
*M0260 46 F7 FE 20 DA 12 03 FE 60 30 2F FE 40 38 02 CB   F.. .....0/.@8..
*M0270 B7 77 CD A1 02 20 1C CD AE 02 DD CB 00 4E 20 10   ..... .......N .
*M0280 DD 23 DD 23 DD 6E 00 CB 85 CB 8D DD 66 01 18 03   .#.#............     ①
*M0290 CD 64 03 22 44 F7 DD 22 46 F7 DD E1 C1 D1 E1 F1   ..."D.."F.......
*M02A0 C9 23 DD 7E 02 E6 FC BD C0 DD 7E 03 BC C9 E5 C5   .#..............
*M02B0 D5 3A B1 F7 B7 28 22 DD 5E 00 CB 83 CB 8B DD 56   .:...(".^......V
*M02C0 01 AF ED 52 45 78 B7 20 01 04 1A CD DD 02 CD E3   ...RE.. ........
*M02D0 02 13 10 F6 3E 0D CD E3 02 D1 C1 E1 C9 FE 20 D0   ....>......... .
*M02E0 CB F7 C9 E5 C5 F5 21 B2 F7 CB 7E 20 FC 2A AD F7   ......!.... .*..
*M02F0 F1 77 23 01 00 FF CD 4F 02 20 03 21 B3 F7 22 AD   ..#....O. .!..".
>

>D,20
*M0000 31 00 00 C3 70 00 00 00 C3 84 0A 00 00 00 00 00   1...............
*M0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................     ②
>

>D500
*M0500 55 02 06 4C CD ED 04 CD F8 04 CD DF 04 2A 2A 2A   U..L.........***     ③
>
```

まています.

図12にPコマンドの実行例を示します. この例では, I/Oポート18Hの内容を読み, それにデータを書き込んでいます. このコンピュータではI/Oポート18Hのリードはdipスイッチであり, ライトはLEDです(図3参照).

したがってI/Oポート18Hにデータを書き込んでもLEDの状態が変化するだけで, 読んだ状態は変りません. この実行例の下2行ではDIPスイッチの設定を変えたので, 読み取ったデータが変化しています.

D(Display Memory)コマンド

>Daaaa, llll CR

コマンドDの後に, 表示すべきメモリのアドレスと, データの長さをカンマ','で区切って入力します. モニタはスクリーンの1行に16バイトのデータを16進数で表示し, その後に各バイトをキャラクタ表示します. キャラクタ表示ができない場合には, ピリオド'.'に変換して表示します.

図13にDコマンドの実行例を示します. 図13の①は, アドレス200Hから始まる100H(256)バイトのメモリの内容を表示しています. ②はアドレスを省略した場合で, この場合にはアドレス0000Hが仮定されます. ③は長さを省略した場合で, この場合には10H(16)が仮定されます.

〈図14〉Aコマンドの実行例

```
>A1234,5678
*68AC
>A89AB,89AB
*1356
>A1234,FFFE
*1232
>
```

〈図15〉Sコマンドの実行例

```
>S1201,1236
*FFCB
>S89AB,1234
*7777
>S0000,5678
*A988
>
```

A(Add Binary Value)コマンド

>Aaaaa, bbbb CR

これは二つの16進数の和を求めるコマンドです. コマンドAの後に和を求めたい二つの16進数を, カンマで区切って入力します. 演算は16ビット加算命令(たとえば, ADD HL, BC)により行われ, 結果のビット15からのキャリーは無視されます. 図14にこのAコマンドの実行例を示します.

S(Subtract Binary Value)コマンド

>Saaaa, bbbb CR

これは二つの16進数の差を求めるためのコマンドです. コマンドSの後に被減数と, 減数とをカンマで区切って16進数で入力します. 図15にこのSコマンドの実行例を示します.

このコマンドは, 相対ジャンプ命令のディスプレースメント部分を求めるときに便利です. たとえば, 1234H番地にあるJR命令の飛び先が1201H番地であったとします. このときには,

>S1201,1236 CR

というコマンドを実行することにより, 両者の差はFFCBHであることがわかり, JR命令のディスプレースメント部分の値はCBHとなります.

F(Fill Memory with Word)コマンド

>Faaaa, llll, dddd CR

これはメモリ・ブロックを指定したデータで埋めるコマンドです. コマンドFに続いて, メモリ・ブロックの先頭のアドレス, 長さ, データの順にそれぞれカンマで区切って入力します.

データは2バイトであり, 0000とすればメモリ・ブロックはすべて00で満たされ, 55AAとすれば, メモリ・ブロックは55 AA 55 AA……といった具合に満たされます.

〈図16〉Fコマンドの実行例

```
>F3000,40,55AA
>D3000,50
*M3000 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA    U.U.U.U.U.U.U.U.   ①
*M3010 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA    U.U.U.U.U.U.U.U.
*M3020 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA    U.U.U.U.U.U.U.U.
*M3030 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA    U.U.U.U.U.U.U.U.
*M3040 DD EF 01 33 CD EE 11 33 CD FF 01 33 CD FE 32 02    ...3...3...3..2.
>

>F3000,40,0
>D3000,50
*M3000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................   ②
*M3010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
*M3020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
*M3030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
*M3040 DD EF 01 33 CD EE 11 33 CD FF 01 33 CD FE 32 02    ...3...3...3..2.
```

〈図 17〉 Vコマンドの実行例

```
>V4000,200,40
>D4000,40
*M4000 ED 4D DB 12 CB 47 20 F4 3E 00 D3 12 3E 40 D3 12   .M...G .>...>@..
*M4010 7E D3 10 3E C0 D3 12 3E 40 D3 12 3A B2 F7 CB F7   ...>...>@..:....
*M4020 32 B2 F7 23 01 00 FF CD 4F 02 20 03 21 B3 F7 22   2..#....O. .!.."
*M4030 AF F7 21 B2 F7 CB BE 18 C3 F5 C5 E5 21 B2 F7 CB   ..!.........!...
>D200,40
*M0200 ED 4D DB 12 CB 47 20 F4 3E 00 D3 12 3E 40 D3 12   .M...G .>...>@..
*M0210 7E D3 10 3E C0 D3 12 3E 40 D3 12 3A B2 F7 CB F7   ...>...>@..:....
*M0220 32 B2 F7 23 01 00 FF CD 4F 02 20 03 21 B3 F7 22   2..#....O. .!.."
*M0230 AF F7 21 B2 F7 CB BE 18 C3 F5 C5 E5 21 B2 F7 CB   ..!.........!...
```

図16にこのFコマンドの実行例を示します．図16の①では，3000H番地からの40H(64)バイトをデータ55AAHで満たしています．Fコマンドの実行結果をDコマンドにより確認しています．図の②は3000H番地からの40Hバイトをゼロ・クリヤする例です．コマンドのデータ部分は"0"だけですが，モニタは0000Hが入力されたと同じ動作をします．

V(Move Memory)コマンド

＞Vaaaa, bbbb, llll CR

これはメモリ・ブロック間の移動を行うコマンドです．コマンドVに続いて，デスティネーション・アドレス，ソース・アドレス，ならびにブロックの長さの順に，それぞれをカンマで区切って入力します．図17にこのコマンドの実行例を示します．

この実行例では，200H番地からの40H(64)バイトのデータを，4000H番地から始まる40Hバイトの領域に移動しています．コマンドの実行結果を確認するために，Dコマンドにより，4000H番地および200H番地から始まる40Hバイトのデータを表示してあります．

C(Compare Memory)コマンド

＞Caaaa, bbbb, llll CR

これは二つのメモリ・ブロックのデータを比較するコマンドです．コマンドCの後に二つのメモリ・ブロックの先頭アドレス，および比較すべきブロックの長さを入力します．図18にこのコマンドの実行例を示します．

この実行例では，まずVコマンドにより0000H番地からの1000Hバイトのデータを，8000H番地から始まる1000Hバイトの領域に移動しています．次にCコマンド(a)により両者を比較しています．両者は等しいはずですので，Cコマンドは何もメッセージを出さず，次のコマンドのためのプロンプトが表示されます．

次にMコマンドを用いて，8000H，8467H番地，ならびに8FFFH番地の内容を変更した後に，前と同じCコマンド(b)を実行しています．変更した3カ所のデータに差異が検出されますので，Cコマンドは差異が見つかったアドレスとデータを表示しています．

・O(Set Offset)コマンド

＞Oaaaa CR

〈図 18〉 Cコマンドの実行例

```
>V8000,0,1000
>C0,8000,1000…… (a)
>M8000
*M8000 31-56
*M8001 00-X
>M8467
*M8467 B7-12
*M8468 C0-X
>M8FFF
*M8FFF 00-EF
*M9000 FF-X
```

```
>C0,8000,1000…… (b)
*0000:31  8000:56
*0467:B7  8467:12
*0FFF:00  8FFF:EF
```

〈図 19〉 Oコマンドの使用例

```
>D400,40
*M0400 3E 20 12 C3 9A 02 E5 44 4D DD 6E 02 CB 85 CB 8D   > .....DM.......
*M0410 DD 66 03 AF ED 42 44 4D E1 C9 01 08 00 EB 09 7D   .....BDM........
*M0420 E6 F8 6F 2B CD A1 02 C2 93 02 DD CB 00 4E C2 9A   ...+.........N..
*M0430 02 DD 23 DD 23 C3 93 02 EB CD 06 04 36 20 0B 79   ..#.#.......6 ..
>O200
>D200,40
*M0200 3E 20 12 C3 9A 02 E5 44 4D DD 6E 02 CB 85 CB 8D   > .....DM.......
*M0210 DD 66 03 AF ED 42 44 4D E1 C9 01 08 00 EB 09 7D   .....BDM........
*M0220 E6 F8 6F 2B CD A1 02 C2 93 02 DD CB 00 4E C2 9A   ...+.........N..
*M0230 02 DD 23 DD 23 C3 93 02 EB CD 06 04 36 20 0B 79   ..#.#.......6 ..
```

これはアドレス・オフセットを設定するコマンドです．コマンドOに続いて，オフセット値を入力します．このオフセット値はメモリを参照するすべてのコマンドに適用されます．

したがってコマンド・レパートリ中のP，A，S，Q以外のコマンドには，すべて適用されます．

モニタはメモリを参照するときには，必ずアドレスにオフセット値を加えたものを，メモリ・アドレスとして用います．リセットから立ち上がると，モニタはオフセット値を0000Hに設定します．

図19にこのOコマンドの使用例を示します．この図ではまずOコマンドを使わずに(つまり，オフセット値＝0000H)，400Hからのデータを40Hバイト表示しています．次にOコマンドによりオフセット値200Hをセットし，200H番地から40Hバイトのデータを表示しています．

図からもわかるように，両Dコマンドにより表示されたデータはまったく等しくなっています．２番目のDコマンドのアドレスは200Hですが，オフセット値が200Hであるため，実際には400H(200H＋200H)番地からのデータが表示されているからです．

このOコマンドを設けた理由は次のとおりです．本

```
>Q
*0000
>Q1234
>Q
*1234
>Q8000
>Q
*8000
```

〈図20〉 Qコマンドの実行例

〈図21〉 Intel Hexフォーマット

: (3AH)	L L	A A A A	0 0	D D D ～ D D	S S

LL：テキスト・レングス(DD…部分の長さ)　DD：テキスト
AA：ロード・アドレス　SS：チェック・サム(LL～SSまでの和がゼロになる)

(a) テキスト・レコード

: (3AH)	0 0	0 0 0 0	0 1	F F

(b) エンド・レコード

〈図22〉 Intel Hexフォーマットのプログラム例

```
:2020000031AC203EC30606D31810FC21522006120E18EDB33EBBD3183E7FD3183EA7D318F2
:2020200006070E18216420EDB21100002100240100041ABE201B13230B78B120F5210024F7
:2020400011012401FF0336FFEDB02161200603188F76C37D0000FF0354025002 80CD002422
:202060008ACFB38700000000000000000000000000000000000000000000000000000000CD
:2020800000000000000000000000000000000000000000000000000000000000000000000040
:0C20A000000000000000000000000000000034
:00000001FF
```

```
                        ;**************************************************
                        ;****            DISPLAY   REGISTERS           ****
                        ;**************************************************
2000    3E 55                   LD      A,55H           ;
2002    01 1234                 LD      BC,1234H        ;
2005    11 5678                 LD      DE,5678H        ;
2008    21 9ABC                 LD      HL,9ABCH        ;
200B    31 0000                 LD      SP,0000H        ;
200E    DD 21 DEF0              LD      IX,0DEF0H       ;
2012    FD 21 ABCD              LD      IY,0ABCDH       ;
2016    B7                      OR      A               ;
2017    08                      EX      AF,AF'          ;
2018    3E 00                   LD      A,00H           ;
201A    B7                      OR      A               ;
201B    D9                      EXX                     ;
201C    01 2345                 LD      BC,2345H        ;
201F    11 6789                 LD      DE,6789H        ;
2022    21 BCDE                 LD      HL,0BCDEH       ;
2025    FF                      RST     38H             ;
                        ;
                                END

>G2000

*PC=2026  SP=FFFE  IX=DEF0  IY=ABCD
*A =00  F =44  B =23  C =45  D =67  E =89  H =BC  L =DE
*A"=55  F"=04  B"=12  C"=34  D"=56  E"=78  H"=9A  L"=BC
```

〈図23〉 RST 38H命令によるレジスタの表示

稿では示しませんが，実はこのモニタにはEPROMを焼く機能があります．Z80の場合，EPROMに焼かれるプログラムは通常ゼロ番地から始まります．このモニタもゼロ番地から始まり，EPROMに収まっています．

したがってEPROMに焼くべきプログラムをメモリ上にロードする場合，オフセットを加える必要があります．どうせオフセット機能を設けるのならば，プログラム・ロードだけではなく，メモリを参照するコマンドすべてに対して，オフセット機能が働くようにしたわけです．

EPROMを焼くプログラムは，2716～27128のすべてをカバーできるようにデザインしたのですが，2764以上のEPROMに関するテストが完全ではないので，残念ながら本稿では割合しました．

・Q(Inquire Offset)

＞Q CR

これは設定されているオフセット値を表示するコマンドです．このコマンドにはパラメータがなく，コマンドQとキャリッジ・リターンのみを入力します．図20にこのQコマンドの実行例を示します．

・L(Load Program)

＞L CR

これはIntel Hex フォーマットのプログラムを，シリアル・インターフェース経由でロードするためのコマンドです．このコマンドにもパラメータはなく，コマンドLとキャリッジ・リターンのみを入力します．図21にIntel Hex フォーマットを示し，図22にこのフォーマットで作られたプログラムの例を示します．

・G(Go to Program)コマンド

＞Gaaaa CR

コマンドGの後に，ジャンプすべきアドレスを入力します．アドレスを省略すると，ゼロ番地へジャンプします．したがって，モニタをリセットしたいような場合には，コマンドGだけを入力します．

レジスタの表示

このモニタでは，RST 38H命令を実行したとき，またはNMIスイッチが押され，NMIが検出されたときに，Z80のレジスタの内容を表示します．図23にRST 38H命令を実行したときのレジスタ表示例を示します．

なお，この図には実行したプログラムも示されています．図24にはNMIにより，レジスタの内容を表示したときのようすを示します．

コントロール・シフト・ファンクション

このモニタでは，キーボードからのコントロール・シフト(ctl)入力に対して，いくつかの機能を持っています．図25にコントロール・シフト・ファンクションの一覧表を示します．

・コントロールA(ctl-A)

カーソルを右に1カラムだけシフトします．カーソルが行の右端にある時は，次の行の左端にきます．スクリーンの最下行の右端にカーソルがあるときには，カーソルは動きません．

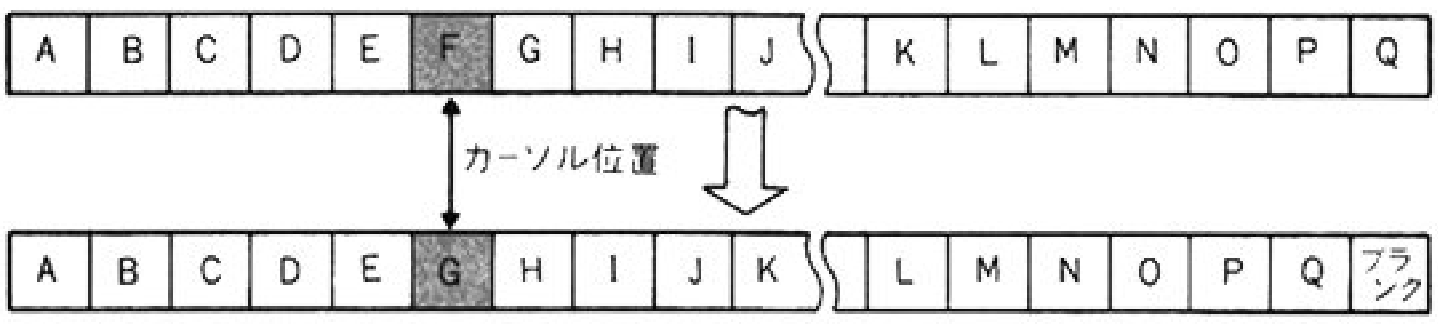
```
*NMI DETECTED
*PC=05D3  SP=FFF6  IX=016C  IY=ABCD
*A =4B  F =4A  B =F6  C =4B  D =07  E =3D  H =F6  L =4B
*A"=55  F"=04  B"=12  C"=34  D"=56  E"=78  H"=9A  L"=BC
>
```

〈図24〉
NMIによるレジスタの表示

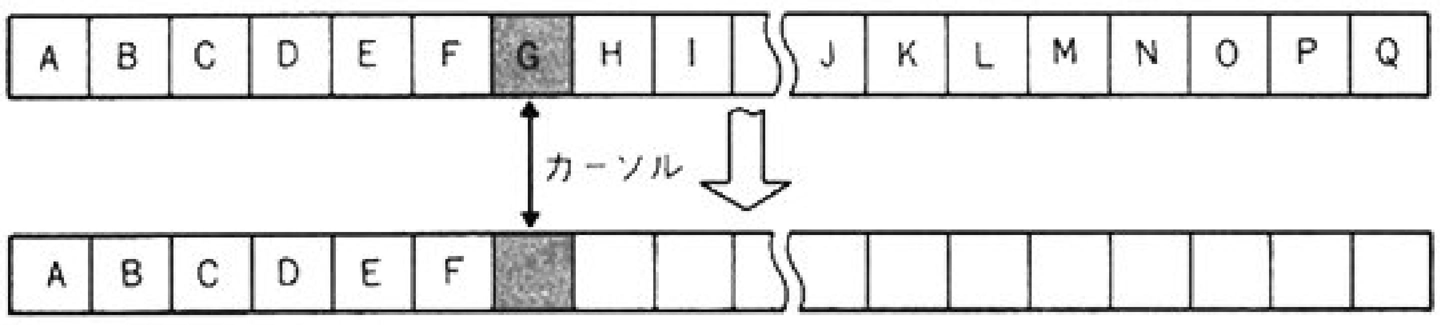
〈図25〉コントロール・シフト・ファンクション

コントロール・シフト	ファンクション名	機能
A	Advance Cursor	カーソルを一つだけ右に進める
B	Back Cursor	カーソルを一つだけ左に進める
C	Clear Screen	スクリーンをクリヤし，カーソルをホーム・ポジションにする
D	Down Cursor	カーソルを1ライン下げる
E	Delete Character	カーソルの下の文字を消去し，同じ行内の文字を左に詰める
L	Erase to End of Line	カーソルより右にある文字を行末まで消去する
P	Printer On/Off	プリンタをON/OFFする
S	Insert Character	カーソルの下にブランクを挿入し，同じ行内の文字を右にシフトする
U	Up Cursor	カーソルを1ライン上げる
W	Duplicate Line	カーソルのある行を次の行にコピーする

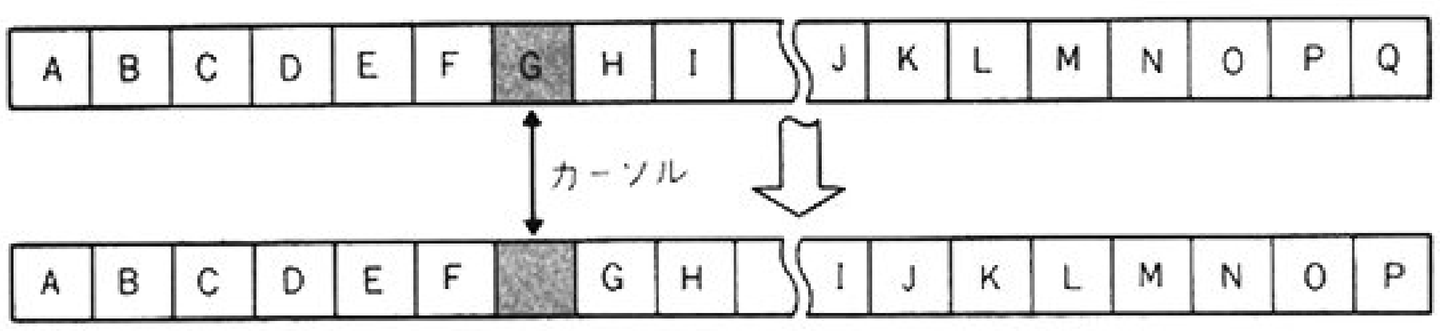
〈図 26〉 コントロールEの機能



〈図 27〉 コントロールLの機能



〈図 28〉 コントロールSの機能



・コントロールB(ctl-B)

カーソルを左に1カラムだけシフトします．カーソルが行の左端にあるときは，上の行の右端にきます．スクリーンの最上位の左端にカーソルがあるときには，カーソルは動きません．

・コントロールC(ctl-C)

スクリーンをクリヤし，カーソルをホーム・ポジション(20行目，カラム1)にセットします．

・コントロールD(ctl-D)

カーソルのカラムは変化せず，行が一つ下がります．カーソルが最下行にあるときは，カーソルは動きません．

・コントロールE(ctl-E)

カーソルの位置の文字を消去し，カーソルより右にある文字列を行内で1カラム左にシフトします．最右端のカラムにはブランクが入ります(図 26)．

・コントロールL(ctl-L)

カーソルのあるカラムから，その行の終りまでの文字をクリヤします(図 27)．

・コントロールS(ctl-S)

カーソルのあるカラムから，その行の終りまでの文字を右に1カラムだけシフトし，カーソル・ポジションにはブランクを挿入します．行の右端の文字は失なわれます．(図 28)．

・コントロールU(ctl-U)

カーソルが1行上に動きます．カーソルが最上行にあるときには，カーソルは動きません．

・コントロールW(ctl-W)

カーソルのある行の内容を次にコピーします．カーソルが最下位にあるときには，この機能は働きません．

・コントロールP(ctl-P)

これまで説明してきた機能は，すべて画面編集用の機能でしたが，このコントロールPはプリンタ制御用です．

リセットから立ち上がると，プリンタはOFF状態にセットされていますが，このコントロールPキーを押すと，プリンタがONになります．再度コントロールPを行うと，プリンタのON/OFFはトグルします．

このモニタは約2Kバイトのプリンタ・バッファを持っており，スクリーンとプリンタは独立に動作できます．スクリーンの動作が待たされるのは，プリンタ・バッファが一杯(Full)になったときだけです．プリンタ・バッファにデータが残っているときに，コントロールPにより，プリンタをOFFすると，プリンタ・バッファはリセットされ，プリント動作は即中断します．

*

以上で製作したシングル・ボード・コンピュータとそのモニタ・プログラムの説明を終りますが，このハードウェアはポート1AHに対してOUT命令を出すと，EPROMをディスエーブルし，RAMをアクティブにする機能があります．

したがって，Vコマンド(V 0，0，1000 CR)により，EPROMの内容をRAMにコピーし，Pコマンドにより，ポート1AHに対して何らかのデータを書き込むと，モニタはRAM上で動くことになります．

この機能を用いると，モニタの一部を変更したりすることもできます．このモニタのデバッグ時には，この機能を十分に活用したものです．

```
                    ;
                    ;       CRTM.MAC        06/12/1983
                    ;***************************************************
                    ;***************************************************
                    ;****                                           ****
                    ;****           Z80 CRT MONITOR                 ****
                    ;****                                           ****
                    ;***************************************************
                    ;***************************************************
                    ;
0000'                       ASEG
                    ;
                    ;
                    ;       I/O PORT EQUATES
0000                SIOAD   EQU     00H             ;SIO CHANNEL A DATA
0001                SIOAC   EQU     SIOAD+1         ;SIO CHANNEL A CONTROL
0002                SIOBD   EQU     SIOAC+1         ;SIO CHANNEL B DATA
0003                SIOBC   EQU     SIOBD+1         ;SIO CHANNEL B CONTROL
                    ;
0004                DMA     EQU     04H             ;Z80-DMA
                    ;
0008                CTC0    EQU     08H             ;CTC CHANNEL 0
0009                CTC1    EQU     CTC0+1          ;CTC CHANNEL 1
000A                CTC2    EQU     CTC1+1          ;CTC CHANNEL 2
000B                CTC3    EQU     CTC2+1          ;CTC CHANNEL 3
                    ;
000C                PIO0AD  EQU     0CH             ;PIO CHANNEL A DATA
000D                PIO0AC  EQU     PIO0AD+1        ;PIO CHANNEL A CONTROL
000E                PIO0BD  EQU     PIO0AC+1        ;PIO CHANNEL B DATA
000F                PIO0BC  EQU     PIO0BD+1        ;PIO CHANNEL B CONTROL
                    ;
0010                PIO1AD  EQU     10H             ;PIO CHANNEL A DATA
0011                PIO1AC  EQU     PIO1AD+1        ;PIO CHANNEL A CONTROL
0012                PIO1BD  EQU     PIO1AC+1        ;PIO CHANNEL B DATA
0013                PIO1BC  EQU     PIO1BD+1        ;PIO CHANNEL B CONTROL
                    ;
0018                DIPSW   EQU     18H             ;DIP SWITCH (READ ONLY)
0018                LED     EQU     18H             ;LED'S (WRITE ONLY)
0019                CNMI    EQU     19H             ;CLEAR NMI FF (WRITE)
001A                DISROM  EQU     1AH             ;DISABLE ROM (WRITE)
001B                CRTRS   EQU     1BH             ;6845 REG. SEL (WRITE)
001C                CLRRDY  EQU     1CH             ;CLEAR DMA READY (WR)
                    ;
                    ;       CODE EQUATES
000D                CR      EQU     0DH             ;CARRIAGE RETURN
                    ;
                    ;       MEMORY EQUATES
0000                STACK   EQU     0000H           ;STACK FRAME
                    ;
F000                SCREEN  EQU     0F000H          ;SCREEN MEMORY
0640                SC$SZ   EQU     80*20           ;SCREEN SIZE
                    ;
F640                KBUF    EQU     SCREEN+SC$SZ    ;KEYBOARD BUFFER
F740                KBUFE   EQU     KBUF+256        ;KBUF END ADDRESS
F740                KBUFIP  EQU     KBUFE           ;KBUF INPUT POINTER
F742                KBUFOP  EQU     KBUFIP+2        ;KBUF OUTPUT POINTER
F744                CURSOR  EQU     KBUFOP+2        ;CURSOR ADDRESS
F746                RTBLP   EQU     CURSOR+2        ;ROW TABLE POINTER
F748                KBLKF   EQU     RTBLP+2         ;KEYBOARD LOCK FLAG
F749                LINEB   EQU     KBLKF+1         ;KEYBOARD LINE BUFFER
F799                PAR0    EQU     LINEB+80        ;PARAMETER 0
```

```
F79B                 PAR1     EQU   PAR0+2         ;PARAMETER 1
F79D                 PAR2     EQU   PAR1+2         ;PARAMETER 2
F79F                 RETS     EQU   PAR2+2         ;ROUTINE ADRS SAVE AREA
F7A1                 HLSAV    EQU   RETS+2         ;HL SAVE AREA
F7A3                 SPSAV    EQU   HLSAV+2        ;SP SAVE AREA
F7A5                 PCSAV    EQU   SPSAV+2        ;PC SAVE AREA
F7A7                 OFSET    EQU   PCSAV+2        ;OFFET VALUE
F7A9                 HLSAVO   EQU   OFSET+2        ;HL SAVE AREA
F7AB                 BCSAVO   EQU   HLSAVO+2       ;BC SAVE AREA
                     ;
F7AD                 PBUFIP   EQU   BCSAVO+2       ;PBUF INPUT POINTER
F7AF                 PBUFOP   EQU   PBUFIP+2       ;PBUF OUTPUT POINTER
F7B1                 PFLAG    EQU   PBUFOP+2       ;PRINT FLAG
F7B2                 PWFLG    EQU   PFLAG+1        ;PRINTER WAIT FLAG
                     ;        BIT   7              ;PBUF FULL INDICATOR
                     ;        BIT   6              ;PRINTER BUSY FLAG
F7B3                 PBUF     EQU   PWFLG+1        ;PRINT BUFFER
FF00                 PBUFE    EQU   STACK-256      ;PBUF END
                     ;
                     ;
0000    31 0000      START:   LD    SP,STACK       ;SET STACK POINTER
0003    C3 0070               JP    STARTO         ;
                     ;
                              ORG   START+08H
                     ;        RST   08H
0008    C3 0A86               JP    RST08          ;
                     ;
                              ORG   START+38H
                     ;        RST   38H
0038    C3 085F               JP    RDISP          ;DISPLAY REGISTERS
                     ;
                     ;
                              ORG   START+40H
                     ;        INTERRUPT VEVTOR
0040    0000         DMAV:    DW    0              ;
0042    0000                  DW    0              ;
0044    0181                  DW    DMAINT         ;END OF BLOCK INT
0046    0000                  DW    0              ;
                     ;
0048    0058         CTCV:    DW    SPINT          ;CTC 0 INTERRUPT
004A    0170                  DW    CTC1I          ;CTC 1 INTERRUPT
004C    0058                  DW    SPINT          ;CTC 2 INTERRUPT
004E    0058                  DW    SPINT          ;CTC 3 INTERRUPT
                     ;
0050    01B8         PIOV:    DW    PIO0AI         ;PIO 0 PORT A INT
0052    0058                  DW    SPINT          ;PIO 0 PORT B INT
0054    0239                  DW    PIO1AI         ;PIO 1 PORT A INT
0056    0058                  DW    SPINT          ;PIO 1 PORT B INT
                     ;
                     ;
                     ;        SPURIOUS INTERRUPT
0058    FB           SPINT:   EI                   ;RE-ENABLE INTERRUPT
0059    ED 4D                 RETI                 ;
                     ;
                     ;        NMI
                              ORG   START+66H
0066    C3 0A69               JP    NMI            ;
                     ;
                     ;
                              ORG   START+70H
0070    3E 18        STARTO:  LD    A,18H          ;RESET SIO CHANNEL A
0072    D3 01                 OUT   (SIOAC),A      ;
0074    D3 03                 OUT   (SIOBC),A      ;RESET SIO CHANNEL B
0076    21 F640               LD    HL,KBUF        ;INITIALIZE KBUFIP
0079    22 F740               LD    (KBUFIP),HL    ;
007C    22 F742               LD    (KBUFOP),HL    ;INITIALIZE KBUFOP
```

```
007F    21 F7B3              LD      HL,PBUF          ;INITIALIZE PBUFIP
0082    22 F7AD              LD      (PBUFIP),HL      ;
0085    22 F7AF              LD      (PBUFOP),HL      ;INITIALIZE PBUFOP
0088    3E 00                LD      A,0              ;CLEAR PRINT FLAG
008A    32 F7B1              LD      (PFLAG),A        ;
008D    32 F7B2              LD      (PWFLG),A        ;CLEAR PWFLG
0090    32 F748              LD      (KBLKF),A        ;CLEAR KBLKF
0093    21 F5F0              LD      HL,SCREEN+80*19  ;INITIALIZE CURSOR
0096    22 F744              LD      (CURSOR),HL      ;
0099    21 016C              LD      HL,RTBL+2*19     ;SET RTBL POINTER
009C    22 F746              LD      (RTBLP),HL       ;
                    ;        NOW CLEAR SCREEN
009F    11 F001              LD      DE,SCREEN+1      ;
00A2    21 F000              LD      HL,SCREEN        ;
00A5    01 063F              LD      BC,SC$SZ-1       ;
00A8    36 20                LD      (HL),20H         ;
00AA    ED B0                LDIR                     ;
                    ;        INITIALIZE PIO 0 PORT B
00AC    3E 0F                LD      A,0FH            ;MODE 0 (OUTPUT)
00AE    D3 0F                OUT     (PIO0BC),A       ;
00B0    3E 07                LD      A,07H            ;DISABLE INTERRUPT
00B2    D3 0F                OUT     (PIO0BC),A       ;
                    ;        INITIALIZE CRT CONTROLLER
00B4    06 00                LD      B,0              ;SET REGISTER NUMBER
00B6    0E 0E                LD      C,PIO0BD         ;LOAD PIO0BD ADDRESS
00B8    21 0126              LD      HL,CRTCMD        ;LOAD COMMAND ADDRESS
00BB    3E 00       START1:  LD      A,0              ;SELECT REGISTER
00BD    D3 1B                OUT     (CRTRS),A        ;
00BF    ED 41                OUT     (C),B            ;LOAD REGISTER NUMBER
00C1    3E 01                LD      A,01H            ;SELECT REGISTER
00C3    D3 1B                OUT     (CRTRS),A        ;
00C5    7E                   LD      A,(HL)           ;LOAD COMMAND
00C6    ED 79                OUT     (C),A            ;
00C8    04                   INC     B                ;BUMP REGISTER NUMBER
00C9    23                   INC     HL               ;BUMP COMMAND POINTER
00CA    78                   LD      A,B              ;END OF COMMAND ?
00CB    FE 10                CP      10H              ;
00CD    20 EC                JR      NZ,START1        ;NO,START1
                    ;        INITIALIZE DMA CONTROLLER
00CF    3E C3                LD      A,0C3H           ;RESET DMA COMMAND
00D1    06 06                LD      B,6              ;SET COUNT
00D3    D3 04       START2:  OUT     (DMA),A          ;RESET DMA
00D5    10 FC                DJNZ    START2           ;6 TIMES ? NO,START2
00D7    06 10                LD      B,DMACL-DMACC    ;LOAD COMMAND LENGTH
00D9    0E 04                LD      C,DMA            ;LOAD DMA ADDRESS
00DB    21 0136              LD      HL,DMACC         ;LOAD COMMAND ADDRESS
00DE    ED B3                OTIR                     ;LOAD COMMAND TO DMA
                    ;        INITIALIZE PIO 0 PORT A FOR KEYBOARD
00E0    3E 4F                LD      A,4FH            ;MODE 1 (INPUT)
00E2    D3 0D                OUT     (PIO0AC),A       ;
00E4    3E 87                LD      A,87H            ;ENABLE INTERRUPT
00E6    D3 0D                OUT     (PIO0AC),A       ;
00E8    3E 50                LD      A,PIOV+0-START   ;LOAD INTERRUPT VECTOR
00EA    D3 0D                OUT     (PIO0AC),A       ;
                    ;        INITIALIZE CTC CHANNEL 1 FOR PRINTER
00EC    3E 03                LD      A,03H            ;RESET CTC CHANNEL 0
00EE    D3 09                OUT     (CTC1),A         ;
00F0    3E A5                LD      A,0A5H           ;ENABLE INTERRUPT
                                                      ;TIMER MODE
                                                      ;PRESCALER=256
                                                      ;TIME CONSTANT FOLLOWS
00F2    D3 09                OUT     (CTC1),A         ;
00F4    3E FF                LD      A,255            ;TIME CONSTANT
00F6    D3 09                OUT     (CTC1),A         ;
00F8    3E 48                LD      A,CTCV+0-START   ;INTERRUPT VECTOR
00FA    D3 08                OUT     (CTC0),A         ;
```

```
                       ;       INITIALIZE PIO 1 FOR PRINTER
00FC    3E OF                  LD      A,0FH           ;PORT A IS OUTPUT MODE
00FE    D3 11                  OUT     (PIO1AC),A      ;
0100    3E 54                  LD      A,PIOV+4-START  ;INTERRUPT VECTOR
0102    D3 11                  OUT     (PIO1AC),A      ;
0104    3E 87                  LD      A,87H           ;ENABLE INTERRUPT
0106    D3 11                  OUT     (PIO1AC),A      ;
0108    3E CF                  LD      A,0CFH          ;PORT B IS BIT MODE
010A    D3 13                  OUT     (PIO1BC),A      ;
010C    3E 3F                  LD      A,3FH           ;SET I/O WORD
010E    D3 13                  OUT     (PIO1BC),A      ;
0110    3E 07                  LD      A,07H           ;DISABLE INTERRUPT
0112    D3 13                  OUT     (PIO1BC),A      ;
                       ;       NOW ENABLE DMA AND INTERRUPT
0114    D3 1C                  OUT     (CLRRDY),A      ;CLEAR READY FLAG
0116    3E 87                  LD      A,87H           ;ENABLE DMA
0118    D3 04                  OUT     (DMA),A         ;
011A    21 0000                LD      HL,START        ;LOAD START ADDRESS
011D    7C                     LD      A,H             ;LOAD UPPER HALF OF
                                                       ;VECTOR ADDRESS
011E    ED 47                  LD      I,A             ;LOAD IT TO IR
0120    ED 5E                  IM      2               ;SET MODE 2 INTERRUPT
0122    FB                     EI                      ;ENABLE INTERRUPT
0123    C3 04FD                JP      BEGIN           ;
                       ;
                       ;       CRTC COMMAND CHAIN
0126    70             CRTCMD: DB      113-1           ;R0
0127    50                     DB      80              ;R1
0128    5C                     DB      92              ;R2
0129    08                     DB      08              ;R3
012A    1A                     DB      26              ;R4
012B    02                     DB      2               ;R5
012C    14                     DB      20              ;R6
012D    17                     DB      23              ;R7
012E    00                     DB      0               ;R8
012F    09                     DB      09              ;R9
0130    60                     DB      60H             ;R10
0131    09                     DB      09H             ;R11
0132    00                     DB      0               ;R12
0133    00                     DB      0               ;R13
0134    00                     DB      0               ;R14
0135    00                     DB      0               ;R15
                       ;
                       ;       DMA COMMAND CHAIN
0136    7D             DMACC:  DB      7DH     ;WR0    TRANSFER
                                               ;       PORT A --> PORT B
                                               ;       PORT A ADDRESS FOLLOWS
                                               ;       BLOCK LENGTH FOLLOWS
0137    F000                   DW      SCREEN  ;       PORT A ADDRESS
0139    063F                   DW      SC$SZ-1 ;       BLOCK LENGTH
013B    14                     DB      14H     ;WR1    PORT A INCREMENT ADRS
                                               ;       PORT A IS MEMORY
013C    58                     DB      58H     ;WR2    PORT B IS I/O
                                               ;       PORT B INCREMENT ADRS
                                               ;       TIMING BYTE FOLLOWS
013D    FE                     DB      0FEH    ;       TWO CYCLES
013E    A0                     DB      0A0H    ;WR3    ENABLE INTERRUPT
013F    DD                     DB      0DDH    ;WR4    PORT B ADDRESS FOLLOWS
                                               ;       INTERRUPT CONTROL BYTE
                                               ;       FOLLOWS
                                               ;       BURST MODE
0140    0000                   DW      0       ;       PORT B ADDRESS
0142    32                     DB      32H     ;       INTERRUPT CONTROL BYTE
                                               ;       INT ON END OF BLOCK
                                               ;       INT VECTOR FOLLOWS
                                               ;       STATUS AFFECTS VECTOR
```

```
0143     40                    DB    DMAV-START   ;       INTERRUPT VECTOR
0144     82                    DB      82H        ;WR5    RDY ACTIVE LOW
                                                  ;       CE/ ONLY
                                                  ;       TERMINATE WHEN EOB
0145     CF                    DB      OCFH       ;WR6    LOAD COMMAND
0146               DMACL       EQU     $
                   ;
                   ;           ROW TABLE
0146     F001      RTBL:       DW      SCREEN+1           ;ROW 0
0148     F050                  DW      SCREEN+80*1        ;ROW 1
014A     F0A0                  DW      SCREEN+80*2        ;ROW 2
014C     F0F0                  DW      SCREEN+80*3        ;ROW 3
014E     F140                  DW      SCREEN+80*4        ;ROW 4
0150     F190                  DW      SCREEN+80*5        ;ROW 5
0152     F1E0                  DW      SCREEN+80*6        ;ROW 6
0154     F230                  DW      SCREEN+80*7        ;ROW 7
0156     F280                  DW      SCREEN+80*8        ;ROW 8
0158     F2D0                  DW      SCREEN+80*9        ;ROW 9
015A     F320                  DW      SCREEN+80*10       ;ROW 10
015C     F370                  DW      SCREEN+80*11       ;ROW 11
015E     F3C0                  DW      SCREEN+80*12       ;ROW 12
0160     F410                  DW      SCREEN+80*13       ;ROW 13
0162     F460                  DW      SCREEN+80*14       ;ROW 14
0164     F4B0                  DW      SCREEN+80*15       ;ROW 15
0166     F500                  DW      SCREEN+80*16       ;ROW 16
0168     F550                  DW      SCREEN+80*17       ;ROW 17
016A     F5A0                  DW      SCREEN+80*18       ;ROW 18
016C     F5F2                  DW      SCREEN+80*19+2     ;ROW 19
016E     F640                  DW      SCREEN+80*20       ;DUMMY ROW
                   ;
                   ;           CTC CHANNEL 1 INTERRUPT
0170     F5        CTC1I:      PUSH    AF                 ;SAVE PSW
0171     3A F7B2               LD      A,(PWFLG)          ;LOAD PRINTER CTL FLAG
0174     CB 77                 BIT     6,A                ;PRINTER BUSY ?
0176     28 04                 JR      Z,CTC1IO           ;NO,CTC1IO
0178     F1                    POP     AF                 ;RESTORE PSW
0179     FB                    EI                         ;RE-ENABLE INTERRUPT
017A     ED 4D                 RETI                       ;
017C     C5        CTC1IO:     PUSH    BC                 ;SAVE BC
017D     E5                    PUSH    HL                 ;SAVE HL
017E     C3 01E9               JP      PRTI               ;GO AND INTITIATE PRINT
                   ;
                   ;           DMA INTERRUPT
0181     F5        DMAINT:     PUSH    AF                 ;SAVE PSW
0182     C5                    PUSH    BC                 ;SAVE BC
0183     E5                    PUSH    HL                 ;SAVE HL
0184     D3 1C                 OUT     (CLRRDY),A         ;CLEAR READY FF
0186     3E 8B                 LD      A,8BH              ;CLEAR DMA STATUS
0188     D3 04                 OUT     (DMA),A            ;
018A     3E CF                 LD      A,0CFH             ;RE-LOAD COMMAND
018C     D3 04                 OUT     (DMA),A            ;
018E     3E 87                 LD      A,87H              ;RE-ENABLE DMA COMMAND
0190     D3 04                 OUT     (DMA),A            ;
                   ;           UPDATE CURSOR
0192     2A F744               LD      HL,(CURSOR)        ;LOAD CURSOR POSITION
0195     7C                    LD      A,H                ;LOAD UPPER CURSOR
0196     E6 0F                 AND     0FH                ;CLEAR UNUSED BITS
0198     67                    LD      H,A                ;
0199     06 0E                 LD      B,14               ;SET REGISTER 14
019B     4C                    LD      C,H                ;UPPER CURSOR TO C
019C     CD 01AA               CALL    SETC               ;SET CURSOR
019F     04                    INC     B                  ;SET REGISTER 15
01A0     4D                    LD      C,L                ;LOWER CURSOR TO C
01A1     CD 01AA               CALL    SETC               ;SET CURSOR
01A4     E1                    POP     HL                 ;RESTORE HL
01A5     C1                    POP     BC                 ;RESTORE BC
```

```
01A6    F1                       POP     AF              ;RESTORE PSW
01A7    FB                       EI                      ;RE-ENABLE INTERRUPT
01A8    ED 4D                    RETI                    ;
                       ;         SET CURSOR SUBROUTINE
                       ;         B=REGISTER NUMBER
                       ;         C=CURSOR ADDRESS
01AA    AF             SETC:     XOR     A               ;TO SELECT REGISTER
01AB    D3 1B                    OUT     (CRTRS),A       ;
01AD    78                       LD      A,B             ;LOAD REGISTER NUMBER
01AE    D3 0E                    OUT     (PIO0BD),A      ;
01B0    3E 01                    LD      A,1             ;TO ACCESS REGISTER
01B2  . D3 1B                    OUT     (CRTRS),A       ;
01B4    79                       LD      A,C             ;LOAD COMMAND
01B5    D3 0E                    OUT     (PIO0BD),A      ;
01B7    C9                       RET                     ;
                       ;         KEYBOARD INTERRUPT
01B8    F5             PIO0AI:   PUSH    AF              ;SAVE PSW
01B9    C5                       PUSH    BC              ;SAVE BC
01BA    E5                       PUSH    HL              ;SAVE HL
01BB    DB 0C                    IN      A,(PIO0AD)      ;READ KEYBOARD DATA
01BD    2F                       CPL                     ;INVERT DATA
01BE    E6 7F                    AND     7FH             ;CLEAR UNUSED BIT
01C0    FE 10                    CP      10H             ;CONTROL P ?
01C2    28 20                    JR      Z,PIOA1         ;YES,PIOA1
01C4    4F                       LD      C,A             ;SAVE IT
01C5    3A F748                  LD      A,(KBLKF)       ;KEYBOARD LOCKED ?
01C8    B7                       OR      A               ;
01C9    20 13                    JR      NZ,PIOAX        ;YES,PIO0AX
                       ;         NOW QUEUE THE KEYBOARD DATA
01CB    2A F740                  LD      HL,(KBUFIP)     ;LOAD KBUF INPUT POINTER
01CE    71                       LD      (HL),C          ;ENQUE THE DATA
01CF    23                       INC     HL              ;BUMP KBUFIP
01D0    01 F740                  LD      BC,KBUFE        ;LOAD KBUF END ADDRESS
01D3    CD 024F                  CALL    CHLBC           ;CHECK IF END OF KBUF
01D6    20 03                    JR      NZ,PIOA0        ;JUMP IF NOT END OF KBUF
01D8    21 F640                  LD      HL,KBUF         ;RESET KBUF POINTER
01DB    22 F740        PIOA0:    LD      (KBUFIP),HL     ;UPDATE KBUFIP
01DE    E1             PIOAX:    POP     HL              ;RESTORE HL
01DF    C1                       POP     BC              ;RESTORE BC
01E0    F1                       POP     AF              ;RESTORE PSW
01E1    FB                       EI                      ;RE-ENABLE INTERRUPT
01E2    ED 4D                    RETI                    ;
01E4    CD 0460        PIOA1:    CALL    CTLPX           ;PROCESS CONTROL P
01E7    18 F5                    JR      PIOAX           ;
                       ;
                       ;         PRINT INITIATION
01E9    ED 4B F7AD     PRTI:     LD      BC,(PBUFIP)     ;LOAD PBUF INPUT POINTER
01ED    2A F7AF                  LD      HL,(PBUFOP)     ;LOAD PBUF OUTPUT PTR
01F0    CD 024F                  CALL    CHLBC           ;BUFFER EMPTY ?
01F3    20 0D                    JR      NZ,PRTIO        ;NO,PRTIO
01F5    3A F7B2                  LD      A,(PWFLG)       ;LOAD PRINTER FLAG
01F8    CB 7F                    BIT     7,A             ;BUFFER FULL ?
01FA    20 06                    JR      NZ,PRTIO        ;YES,PRTIO
01FC    E1             PRTIX:    POP     HL              ;RESTORE HL
01FD    C1                       POP     BC              ;RESTORE BC
01FE    F1                       POP     AF              ;RESTORE AF
01FF    FB                       EI                      ;RE-ENABLE INTERRUPT
0200    ED 4D                    RETI                    ;
0202    DB 12          PRTIO:    IN      A,(PIO1BD)      ;GET PRINTER STATUS
0204    CB 47                    BIT     0,A             ;PRINTER BUSY ?
0206    20 F4                    JR      NZ,PRTIX        ;YES,PRTIX
0208    3E 00                    LD      A,00H           ;CLEAR ACK FF
020A    D3 12                    OUT     (PIO1BD),A      ;
020C    3E 40                    LD      A,40H           ;ENABLE ACK FF
020E    D3 12                    OUT     (PIO1BD),A      ;
0210    7E                       LD      A,(HL)          ;LOAD PRINT DATA
```

```
0211     D3 10                 OUT    (PIO1AD),A       ;LOAD IT TO PIO1AD
0213     3E C0                 LD     A,0C0H           ;ACTIVATE STROBE
0215     D3 12                 OUT    (PIO1BD),A       ;
0217     3E 40                 LD     A,40H            ;NEGATE STROBE
0219     D3 12                 OUT    (PIO1BD),A       ;
021B     3A F7B2               LD     A,(PWFLG)        ;LOAD PRINTER CTL FLAG
021E     CB F7                 SET    6,A              ;SET PRINTER BUSY
0220     32 F7B2               LD     (PWFLG),A        ;
0223     23                    INC    HL               ;BUMP PBUFOP
0224     01 FF00               LD     BC,PBUFE         ;LOAD PBUF END ADDRESS
0227     CD 024F               CALL   CHLBC            ;END OF PBUF ?
022A     20 03                 JR     NZ,PRTI1         ;NO,PRTI1
022C     21 F7B3               LD     HL,PBUF          ;WRUP AROUND POINTER
022F     22 F7AF       PRTI1:  LD     (PBUFOP),HL      ;UPDATE PBUFOP
0232     21 F7B2               LD     HL,PWFLG         ;LOAD PWFLG ADDRESS
0235     CB BE                 RES    7,(HL)           ;CLEAR PBUF FULL
0237     18 C3                 JR     PRTIX            ;
                       ;
                       ;       PRINTER INTERRUPT
0239     F5            PIO1AI: PUSH   AF               ;SAVE PSW
023A     C5                    PUSH   BC               ;SAVE BC
023B     E5                    PUSH   HL               ;SAVE HL
023C     21 F7B2               LD     HL,PWFLG         ;LOAD PWFLG ADDRESS
023F     CB 76                 BIT    6,(HL)           ;PRINTER BUSY ?
0241     28 B9                 JR     Z,PRTIX          ;NO,PRTIX
0243     CB B6                 RES    6,(HL)           ;CLEAR PRINTER BUSY
0245     3E 00                 LD     A,00H            ;CLEAR ACK FF
0247     D3 12                 OUT    (PIO1BD),A       ;
0249     3E 40                 LD     A,40H            ;
024B     D3 12                 OUT    (PIO1BD),A       ;
024D     18 9A                 JR     PRTI             ;
                       ;
                       ;       COMPARE HL AND BC REGISTER
024F     78            CHLBC:  LD     A,B              ;
0250     BC                    CP     H                ;
0251     C0                    RET    NZ               ;
0252     79                    LD     A,C              ;
0253     BD                    CP     L                ;
0254     C9                    RET                     ;
                       ;
                       ;       DISPLAY CHARCTER SUBROUTINE
0255     F5            DISP:   PUSH   AF               ;SAVE PSW
0256     E5                    PUSH   HL               ;SAVE HL
0257     D5                    PUSH   DE               ;SAVE DE
0258     C5                    PUSH   BC               ;SAVE BC
0259     DD E5                 PUSH   IX               ;SAVE IX
025B     2A F744               LD     HL,(CURSOR)      ;LOAD CURSOR ADDRESS
025E     DD 2A F746            LD     IX,(RTBLP)       ;LOAD RTBL POINTER
0262     FE 20                 CP     20H              ;LOWER THAN 20H ?
0264     DA 0312               JP     C,DISPF          ;YES,DISPF
0267     FE 60                 CP     60H              ;GREATER THAN 60H ?
0269     30 2F                 JR     NC,DISPX         ;YES,DISPX
026B     FE 40                 CP     40H              ;GREATER THAN 40H ?
026D     38 02                 JR     C,DISP0          ;NO,DISP0
026F     CB B7                 RES    6,A              ;CLEAR BIT 6
0271     77            DISP0:  LD     (HL),A           ;PUT CHAR. INTO MEMORY
0272     CD 02A1               CALL   CKEOL            ;CHECK IF EOL
0275     20 1C                 JR     NZ,DISP2         ;NO,DISPX
0277     CD 02AE               CALL   QPRTS            ;ENQUEU PRINT DATA
027A     DD CB 00 4E           BIT    1,(IX+0)         ;IS THIS LAST LINE ?
027E     20 10                 JR     NZ,DISP1         ;YES,DISP1
0280     DD 23         DISP0X: INC    IX               ;BUMP RTBL POINTER
0282     DD 23                 INC    IX               ;
0284     DD 6E 00              LD     L,(IX+0)         ;LOAD NEXT LINE ADRS
0287     CB 85                 RES    0,L              ;
0289     CB 8D                 RES    1,L              ;
```

```
028B	DD 66 01		LD	H,(IX+1)	;
028E	18 03			JR	DISP2		;
0290	CD 0364		DISP1:	CALL	ROLLU		;ROLL UP SCREEN
0293	22 F744		DISP2:	LD	(CURSOR),HL	;UPDATE CURSOR
0296	DD 22 F746		LD	(RTBLP),IX	;UPDATE RTBL POINTER
029A	DD E1		DISPX:	POP	IX		;RESTORE IX
029C	C1			POP	BC		;RESTORE BC
029D	D1			POP	DE		;RESTORE DE
029E	E1			POP	HL		;RESTORE HL
029F	F1			POP	AF		;RESTORE PSW
02A0	C9			RET			;
			;	CHECK IF END OF LINE
02A1	23		CKEOL:	INC	HL		;BUMP CURSOR
02A2	DD 7E 02		LD	A,(IX+2)	;LOAD NEXT LINE ADRS
02A5	E6 FC			AND	0FCH		;CLEAR FLAG
02A7	BD			CP	L		;MAY BE EOL ?
02A8	C0			RET	NZ		;NO,RETURN
02A9	DD 7E 03		LD	A,(IX+3)	;LOAD NEXT LINE ADRS
02AC	BC			CP	H		;END OF LINE ?
02AD	C9			RET			;
			;
			;	ENQUE PRINT DATA FROM SCREEN
02AE	E5		QPRTS:	PUSH	HL		;SAVE HL
02AF	C5			PUSH	BC		;SAVE BC
02B0	D5			PUSH	DE		;SAVE BC
02B1	3A F7B1			LD	A,(PFLAG)	;LOAD PRINT FLAG
02B4	B7			OR	A		;NEED PRINT ?
02B5	28 22			JR	Z,QPRTSX	;NO,QPRTSX
02B7	DD 5E 00		LD	E,(IX+0)	;LOAD BOL ADDRESS
02BA	CB 83			RES	0,E		;CLEAR FLAGS
02BC	CB 8B			RES	1,E		;
02BE	DD 56 01		LD	D,(IX+1)	;
02C1	AF			XOR	A		;CLEAR CARRY FOR SBC
02C2	ED 52			SBC	HL,DE		;GET DATA LENGTH
02C4	45			LD	B,L		;LENGTH TO B
02C5	78			LD	A,B		;LENGTH=0 ?
02C6	B7			OR	A		;
02C7	20 01			JR	NZ,QPRTS0	;NO,QPRTS
02C9	04			INC	B		;SET LENGTH=1
02CA	1A		QPRTS0:	LD	A,(DE)		;LOAD DATA FROM SCREEN
02CB	CD 02DD			CALL	CONA		;CONVERT TO ASCCII
02CE	CD 02E3			CALL	QPDT		;ENQUE PRINT DATA
02D1	13			INC	DE		;BUMP DATA POINTER
02D2	10 F6			DJNZ	QPRTS0		;ALL QUEUED ? NO,QPRTS0
02D4	3E 0D			LD	A,CR		;LOAD CR CODE
02D6	CD 02E3			CALL	QPDT		;ENQUE CR CODE
02D9	D1		QPRTSX:	POP	DE		;RESTORE DE
02DA	C1			POP	BC		;RESTORE BC
02DB	E1			POP	HL		;RESTORE HL
02DC	C9			RET			;
			;
			;	CONVERT SCREEN DATA TO ASCII
02DD	FE 20		CONA:	CP	20H		;LOWER THAN 20H ?
02DF	D0			RET	NC		;NO,RETURN
02E0	CB F7			SET	6,A		;GET ASCII CODE
02E2	C9			RET			;
			;
			;	ENQUE PRINT DATA INTO PBUF
02E3	E5		QPDT:	PUSH	HL		;SAVE HL
02E4	C5			PUSH	BC		;SAVE BC
02E5	F5			PUSH	AF		;SAVE DATA TO BE QUEUED
02E6	21 F7B2			LD	HL,PWFLG	;LOAD PWFLG ADDRESS
02E9	CB 7E		QPDT0:	BIT	7,(HL)		;PBUF FUL ?
02EB	20 FC			JR	NZ,QPDT0	;YES,QPDT0
02ED	2A F7AD			LD	HL,(PBUFIP)	;LOAD PBUF INPUT POINTER
02F0	F1			POP	AF		;RESTORE DATA
```

```
02F1    77                    LD      (HL),A          ;ENQUE DATA
02F2    23                    INC     HL              ;
02F3    01 FF00               LD      BC,PBUFE        ;LOAD PBUF END ADDRESS
02F6    CD 024F               CALL    CHLBC           ;END OF PBUF ?
02F9    20 03                 JR      NZ,QPDT1        ;NO,QPDT1
02FB    21 F7B3               LD      HL,PBUF         ;WRUP AROUND PBUF PTR
02FE    22 F7AD       QPDT1:  LD      (PBUFIP),HL     ;UPDATE PBUFIP
0301    ED 4B F7AF            LD      BC,(PBUFOP)     ;LOAD PBUF OUTPUT PTR
0305    CD 024F               CALL    CHLBC           ;PBUF FULL ?
0308    20 05                 JR      NZ,QPDT2        ;NO,QPDT2
030A    21 F7B2               LD      HL,PWFLG        ;LOAD PWFLG ADDRESS
030D    CB FE                 SET     7,(HL)          ;SET PBUF FULL
030F    C1            QPDT2:  POP     BC              ;RESTORE BC
0310    E1                    POP     HL              ;RESTORE HL
0311    C9                    RET                     ;
                      ;
                      ;       FUNCTION CODE
0312    4F            DISPF:  LD      C,A             ;CODE TO C
0313    06 00                 LD      B,0             ;CLEAR B
0315    EB                    EX      DE,HL           ;CUSOR ADDRESS TO DE
0316    21 0324               LD      HL,FCTBL        ;LOAD FCTBL ADDRESS
0319    09                    ADD     HL,BC           ;GET TABLE ENTRY
031A    09                    ADD     HL,BC           ;
031B    7E                    LD      A,(HL)          ;LOAD ROUTINE ADDRESS
031C    23                    INC     HL              ;BUMP TABLE POINTER
031D    66                    LD      H,(HL)          ;LOAD ROUTINR ADDRESS
031E    6F                    LD      L,A             ;SAVE LOWER ADDRESS
031F    B4                    OR      H               ;NULL ENTRY ?
0320    CA 029A               JP      Z,DISPX         ;YES,DISPX
0323    E9                    JP      (HL)            ;GO TO EACH ROUTINE
                      ;
                      ;       FUNCTION ROUTINE ADDRESS TABLE
0324    0000          FCTBL:  DW      0               ;CONTROL @
0326    0386                  DW      CTLA            ;CONTROL A
0328    0398                  DW      CTLB            ;CONTROL B
032A    03B7                  DW      CTLC            ;CONTROL C
032C    03D9                  DW      CTLD            ;CONTROL D
032E    03F2                  DW      CTLE            ;CONTROL E
0330    0000                  DW      0               ;CONTROL F
0332    0000                  DW      0               ;CONTROL G
0334    0398                  DW      CTLB            ;CONTROL H
0336    041A                  DW      CTLI            ;CONTROL I
0338    0000                  DW      0               ;CONTROL J
033A    0000                  DW      0               ;CONTROL K
033C    0438                  DW      CTLL            ;CONTROL L
033E    044B                  DW      CRRT            ;CONTROL M
0340    0000                  DW      0               ;CONTROL N
0342    0000                  DW      0               ;CONTROL O
0344    0459                  DW      CTLP            ;CONTROL P
0346    0000                  DW      0               ;CONTROL Q
0348    0000                  DW      0               ;CONTROL R
034A    0476                  DW      CTLS            ;CONTROL S
034C    0000                  DW      0               ;CONTROL T
034E    048D                  DW      CTLU            ;CONTROL U
0350    0000                  DW      0               ;CONTROL V
0352    04C1                  DW      CTLW            ;CONTROL W
0354    0000                  DW      0               ;CONTROL X
0356    0000                  DW      0               ;CONTROL Y
0358    0000                  DW      0               ;CONTROL Z
035A    0000                  DW      0               ;
035C    0000                  DW      0               ;
035E    0000                  DW      0               ;
0360    0000                  DW      0               ;
0362    0000                  DW      0               ;
                      ;
```

```
                                ;       ROLL UP SCREEN
0364    C5              ROLLU:  PUSH    BC              ;SAVE BC
0365    D5                      PUSH    DE              ;SAVE DE
0366    21 F050                 LD      HL,SCREEN+80    ;LOAD 'FROM' ADDRESS
0369    11 F000                 LD      DE,SCREEN       ;LOAD 'TO' ADDRESS
036C    01 05F0                 LD      BC,SC$SZ-80     ;LOAD COUNT
036F    ED B0                   LDIR                    ;DO ROLLING UP
                                ;       CLEAR LAST LINE
0371    21 F5F0                 LD      HL,SCREEN+80*19 ;LOAD 'FROM' ADDRESS
0374    13                      INC     DE              ;SET 'TO' ADDRESS
0375    01 004F                 LD      BC,80-1         ;SET COUNT
0378    36 20                   LD      (HL),20H        ;SET BLANK CODE
037A    ED B0                   LDIR                    ;DO CLEARING
037C    21 F5F0                 LD      HL,SCREEN+80*19 ;SET CURSOR ADDRESS
037F    DD 21 016C              LD      IX,RTBL+2*19    ;SET RTBL POINTER
0383    D1                      POP     DE              ;RESTORE DE
0384    C1                      POP     BC              ;RESTORE BC
0385    C9                      RET                     ;
                                ;
                                ;
                                ;       CONTROL A (ADVANCE CURSOR)
0386    EB              CTLA:   EX      DE,HL           ;CURSOR TO HL
0387    CD 02A1                 CALL    CKEOL           ;CHECK IF END OF LINE
038A    C2 0293                 JP      NZ,DISP2        ;JUMP IF NOT EOL
038D    DD CB 00 4E             BIT     1,(IX+0)        ;WAS THIS LAST LINE ?
0391    CA 0280                 JP      Z,DISPOX        ;NO,DISPOX
0394    2B                      DEC     HL              ;DONT MOVE CURSOR
0395    C3 029A                 JP      DISPX           ;
                                ;
                                ;       CONTROL B (BACK CURSOR)
0398    EB              CTLB:   EX      DE,HL           ;CURSOR TO HL
0399    DD 4E 00                LD      C,(IX+0)        ;LOAD BOL ADDRESS
039C    CB 81                   RES     0,C             ;CLEAR FLAGS
039E    CB 89                   RES     1,C             ;
03A0    DD 46 01                LD      B,(IX+1)        ;
03A3    CD 024F                 CALL    CHLBC           ;BEGINNING OF LINE ?
03A6    20 0B                   JR      NZ,CTLBO        ;JUMP IF NOT BOL
03A8    DD CB 00 46             BIT     0,(IX+0)        ;TOP OF LINE ?
03AC    C2 029A                 JP      NZ,DISPX        ;YES,DISPX
03AF    DD 2B                   DEC     IX              ;BACK RTBL POINTER
03B1    DD 2B                   DEC     IX              ;
03B3    2B              CTLBO:  DEC     HL              ;BACK CURSOR
03B4    C3 0293                 JP      DISP2           ;
                                ;
                                ;       CONTROL C (CLEAR SCREEN)
03B7    21 F000         CTLC:   LD      HL,SCREEN       ;LOAD 'FROM' ADDRESS
03BA    11 F001                 LD      DE,SCREEN+1     ;LOAD 'TO' ADDRESS
03BD    01 063F                 LD      BC,SC$SZ-1      ;SET COUNT
03C0    36 20                   LD      (HL),20H        ;CLEAR TOP OF SCREEN
03C2    ED B0                   LDIR                    ;
03C4    CD 03CA                 CALL    HOMEC           ;CURSOR TO HOME
03C7    C3 029A                 JP      DISPX           ;
                                ;
                                ;       CURSOR TO HOME
03CA    21 F5F0         HOMEC:  LD      HL,SCREEN+80*19 ;LOAD HOME ADDRESS
03CD    22 F744                 LD      (CURSOR),HL     ;STORE IT INTO CURSOR
03D0    DD 21 016C              LD      IX,RTBL+2*19    ;LOAD HOME RTBL ADDRESS
03D4    DD 22 F746              LD      (RTBLP),IX      ;STORE IT INTO RTBLP
03D8    C9                      RET                     ;
                                ;
                                ;       CONTROL D (DOWN CURSOR)
03D9    DD CB 00 4E     CTLD:   BIT     1,(IX+0)        ;HERE IS LAST LINE ?
03DD    C2 03EC                 JP      NZ,CTLDO        ;YES,CTLDO
03E0    EB                      EX      DE,HL           ;CURSOR TO HL
03E1    01 0050                 LD      BC,80           ;SET 80
03E4    09                      ADD     HL,BC           ;DOWN CURSOR
03E5    DD 23                   INC     IX              ;BUMP RTBL POINTER
```

```
03E7     DD 23                INC     IX            ;
03E9     C3 0293              JP      DISP2         ;
03EC     CD 0364     CTLDO:   CALL    ROLLU         ;ROLLING UP
03EF     C3 0293              JP      DISP2         ;
                     ;
                     ;        CONTROL E (ERASE CHARACTER)
03F2     EB          CTLE:    EX      DE,HL         ;CURSOR TO HL
03F3     CD 0406              CALL    GETSZ         ;GET SIZE TO BE MOVED
03F6     54                   LD      D,H           ;CURSOR TO DE
03F7     5D                   LD      E,L           ;
03F8     23                   INC     HL            ;ADJUST FOR LDIR
03F9     0B                   DEC     BC            ;AJUST
03FA     78                   LD      A,B           ;LENGTH=0 ?
03FB     B1                   OR      C             ;
03FC     28 02                JR      Z,CTLE0       ;YES,CTLE
03FE     ED B0                LDIR                  ;SIFT LEFT ONE BYTE
0400     3E 20       CTLE0:   LD      A,20H         ;
0402     12                   LD      (DE),A        ;SET BLANK IN TAIL
0403     C3 029A              JP      DISPX         ;
                     ;
                     ;        GET SIZE TO BE MOVED
0406     E5          GETSZ:   PUSH    HL            ;SAVE CURSOR ADDRESS
0407     44                   LD      B,H           ;CURSOR TO BC
0408     4D                   LD      C,L           ;
0409     DD 6E 02             LD      L,(IX+2)      ;LOAD EOL ADDRESS
040C     CB 85                RES     0,L           ;CLEAR FLAGS
040E     CB 8D                RES     1,L           ;
0410     DD 66 03             LD      H,(IX+3)      ;
0413     AF                   XOR     A             ;CLEAR CARRY
0414     ED 42                SBC     HL,BC         ;
0416     44                   LD      B,H           ;LENGTH TO HL
0417     4D                   LD      C,L           ;
0418     E1                   POP     HL            ;
0419     C9                   RET                   ;
                     ;
                     ;        CONTROL I (FORWARD TAB)
041A     01 0008     CTLI:    LD      BC,8          ;SET TAB COUNT
041D     EB                   EX      DE,HL         ;CURSOR TO HL
041E     09                   ADD     HL,BC         ;DO TAB
041F     7D                   LD      A,L           ;LOAD CURSOR LOWER
0420     E6 F8                AND     0F8H          ;GET MULTIPLES OF 8
0422     6F                   LD      L,A           ;
0423     2B                   DEC     HL            ;ADJUSTING
0424     CD 02A1              CALL    CKEOL         ;CHECK IF EOL
0427     C2 0293              JP      NZ,DISP2      ;JUMP IF NOT EOL
042A     DD CB 00 4E          BIT     1,(IX+0)      ;HERE IS LAST LINE ?
042E     C2 029A              JP      NZ,DISPX      ;YES,DISPX
0431     DD 23                INC     IX            ;BUMP RTBL POINTER
0433     DD 23                INC     IX            ;
0435     C3 0293              JP      DISP2         ;
                     ;
                     ;        CONTROL L (ERASE TO END OF LINE)
0438     EB          CTLL:    EX      DE,HL         ;CURSOR TO HL
0439     CD 0406              CALL    GETSZ         ;GET SIZE TO BE ERASED
043C     36 20                LD      (HL),20H      ;CLEAR TOP OF STRING
043E     0B                   DEC     BC            ;ADJUST LENGTH
043F     79                   LD      A,C           ;LENGTH =0 ?
0440     CA 029A              JP      Z,DISPX       ;YES,DISPX
0443     54                   LD      D,H           ;CURSOR TO DE
0444     5D                   LD      E,L           ;
0445     13                   INC     DE            ;ADJUST
0446     ED B0                LDIR                  ;DO ERASING
0448     C3 029A              JP      DISPX         ;
                     ;
                     ;        CONTROL M/CR (CARRIAGE RETURN)
044B     EB          CRRT:    EX      DE,HL         ;CURSOR TO HL
```

```
044C    CD 02AE                 CALL    QPRTS               ;ENQUE PRINT DATA
044F    DD CB 00 4E             BIT     1,(IX+0)            ;HERE IS LAST LINE ?
0453    CA 0280                 JP      Z,DISPOX            ;NO,DISPOX
0456    C3 0290                 JP      DISP1               ;
                        ;
                        ;       CONTROL P (PRINTER ON/OFF)
0459    CD 0460         CTLP:   CALL    CTLPX               ;PROCESS CONTROL P
045C    FB                      EI                          ;ENABLE INTERRUPT
045D    C3 029A                 JP      DISPX               ;
                        ;
                        ;       PROCESS CONTROL P
0460    3A F7B1         CTLPX:  LD      A,(PFLAG)           ;LOAD PRINTER FLAG
0463    2F                      CPL                         ;FLIP THE FLAG
0464    32 F7B1                 LD      (PFLAG),A           ;UPDATE PFLAG
0467    B7                      OR      A                   ;PRINT OFF ?
0468    C0                      RET     NZ                  ;NO,RETURN
0469    F3                      DI                          ;DISABLE INTERRUPT
046A    2A F7AD                 LD      HL,(PBUFIP)         ;RESET PBUF
046D    22 F7AF                 LD      (PBUFOP),HL         ;
0470    21 F7B2                 LD      HL,PWFLG            ;LOAD FLAG ADDRESS
0473    CB BE                   RES     7,(HL)              ;CLEAR BUFFER FULL
0475    C9                      RET                         ;
                        ;
                        ;       CONTROL S (INSERT CHARACTER)
0476    EB              CTLS:   EX      DE,HL               ;CURSOR TO HL
0477    CD 0406                 CALL    GETSZ               ;GET SIZE TO BE MOVED
047A    0B                      DEC     BC                  ;ADJUSTING
047B    79                      LD      A,C                 ;CURSOR IS ON THE LAST ?
047C    B7                      OR      A                   ;
047D    CA 029A                 JP      Z,DISPX             ;YES,DISPX
0480    E5                      PUSH    HL                  ;SAVE CURSOR ADDRESS
0481    09                      ADD     HL,BC               ;HL POINTS EOL
0482    54                      LD      D,H                 ;DE POINTS EOL
0483    5D                      LD      E,L                 ;
0484    2B                      DEC     HL                  ;HL POINTS BEFORE EOL
0485    ED B8                   LDDR                        ;DHIFT RIGHT ONE BYTE
0487    E1                      POP     HL                  ;RESTORE CURSOR
0488    36 20                   LD      (HL),20H            ;FILL SPACE
048A    C3 029A                 JP      DISPX               ;
                        ;
                        ;       CONTROL U (CURSOR UP)
048D    DD CB 00 46     CTLU:   BIT     0,(IX+0)            ;HERE IS TOP OF LINE ?
0491    C2 04A2                 JP      NZ,CTLUO            ;YES,CTLUO
0494    EB                      EX      DE,HL               ;CURSOR TO HL
0495    01 0050                 LD      BC,80               ;SET 80
0498    AF                      XOR     A                   ;CLEAR CARRY
0499    ED 42                   SBC     HL,BC               ;CURSOR UP
049B    DD 2B                   DEC     IX                  ;BACK RTBL POINTER
049D    DD 2B                   DEC     IX                  ;
049F    C3 0293                 JP      DISP2               ;
04A2    CD 04A8         CTLUO:  CALL    ROLLD               ;ROLLING DOWN
04A5    C3 029A                 JP      DISPX               ;
                        ;
                        ;       ROLLING DOWN
04A8    21 F5EF         ROLLD:  LD      HL,SCREEN+80*18+80-1      ;
04AB    11 F63F                 LD      DE,SCREEN+80*19+80-1      ;
04AE    01 05F0                 LD      BC,SC$SZ-80         ;
04B1    ED B8                   LDDR                        ;
04B3    21 F000                 LD      HL,SCREEN           ;
04B6    11 F001                 LD      DE,SCREEN+1         ;
04B9    01 004F                 LD      BC,80-1             ;
04BC    36 20                   LD      (HL),20H            ;
04BE    ED B0                   LDIR                        ;
04C0    C9                      RET                         ;
                        ;
```

```
                          ;         CONTROL W (DUPLICATE LINE)
04C1      DD CB 00 4E     CTLW:     BIT       1,(IX+0)          ;HERE IS LAST LINE ?
04C5      C2 029A                   JP        NZ,DISPX          ;YES,DISPX
04C8      D5                        PUSH      DE                ;SAVE CURSOR ADDRESS
04C9      DD 5E 00                  LD        E,(IX+0)          ;LOAD BOL ADDRESS
04CC      CB 83                     RES       0,E               ;CLEAR FLAG
04CE      DD 56 01                  LD        D,(IX+1)          ;
04D1      21 0050                   LD        HL,80             ;LOAD 80
04D4      19                        ADD       HL,DE             ;GET NEXT LINE ADRS
04D5      EB                        EX        DE,HL             ;
04D6      01 0050                   LD        BC,80             ;LOAD LENGTH TO MOVE
04D9      ED B0                     LDIR                        ;DO DUPLICATION
04DB      D1                        POP       DE                ;CURSOR TO DE
04DC      C3 03D9                   JP        CTLD              ;CURSOR DOWN
                          ;
                          ;
                          ;         MESSAGE OUTPUT ROUTINE
04DF      E3              MGOUT:    EX        (SP),HL           ;GET MESSAGE ADDRESS
04E0      7E              MGOUT0:   LD        A,(HL)            ;LOAD MESSAGE CONTENT
04E1      23                        INC       HL                ;BUMP MESSAGE POINTER
04E2      FE FF                     CP        0FFH              ;MESSAGE END ?
04E4      28 05                     JR        Z,MGOUT1          ;YES,MGOUT1
04E6      CD 0255                   CALL      DISP              ;DISPLAY IT
04E9      18 F5                     JR        MGOUT0            ;GO AND PROCESS NEXT
04EB      E3              MGOUT1:   EX        (SP),HL           ;RETURN ADDRESS INTO STK
04EC      C9                        RET                         ;
                          ;
                          ;         DISPLAY ASTERISCS
                          ;         B=COUNT
04ED      CD 04F3         OUTAR:    CALL      OUTA              ;DISPLAY ONE '*'
04F0      10 FB                     DJNZ      OUTAR             ;DONE ? NO,OUTAR
04F2      C9                        RET                         ;
                          ;
                          ;         DISPLAY ONE ASTERISC
04F3      3E 2A           OUTA:     LD        A,'*'             ;LOAD '*'
04F5      C3 0255                   JP        DISP              ;DISPLAY IT
                          ;
                          ;         OUTPUT CR
04F8      3E 0D           NEWL:     LD        A,CR              ;LOAD CR CODE
04FA      C3 0255                   JP        DISP              ;OUTPUT IT
                          ;
                          ;
                          ;****************************************************
                          ;****                                            ****
                          ;****     HERE IS THE BEGINNING OF THIS MONITOR  ****
                          ;****                                            ****
                          ;****************************************************
04FD      3E 03           BEGIN:    LD        A,03H             ;CLEAR SCREEN
04FF      CD 0255                   CALL      DISP              ;
0502      06 4C                     LD        B,76              ;SET '*' COUNT
0504      CD 04ED                   CALL      OUTAR             ;DISPLAY 76 *'S
0507      CD 04F8                   CALL      NEWL              ;DO CARRIAGE RETURN
050A      CD 04DF                   CALL      MGOUT             ;
050D      2A 2A 2A 2A               DB        '****     Z80 SCREEN MONITOR     VER A    '
0511      20 20 20 20
0515      5A 38 30 20
0519      53 43 52 45
051D      45 4E 20 4D
0521      4F 4E 49 54
0525      4F 52 20 20
0529      20 20 56 45
052D      52 20 41 20
0531      20 20
0533      4A 55 4C 59               DB        'JULY 12TH 1983                      ****'
0537      20 31 32 54
053B      48 20 31 39
```

```
053F     38 33 20 20
0543     20 20 20 20
0547     20 20 20 20
054B     20 20 20 20
054F     20 20 20 20
0553     20 20 2A 2A
0557     2A 2A
0559     0D FF                DB      CR,0FFH
055B     06 4C                LD      B,76            ;DISPLAY 76 *'S
055D     CD 04ED              CALL    OUTAR           ;
0560     CD 04F8              CALL    NEWL            ;DO CARRIAGE RETURN
0563     21 0000              LD      HL,0            ;RESET OFFSET VALUE
0566     22 F7A7              LD      (OFSET),HL      ;
0569     3E 3E       BEGIN0:  LD      A,'>'           ;SET PROMPT
056B     CD 0255              CALL    DISP            ;DISPLAY IT ONTO SCREEN
056E     CD 057A              CALL    RLINE           ;READ COMMAND LINE
0571     DD 21 06B7           LD      IX,CTBL         ;LOAD CTBL ADDRESS
0575     CD 05F1              CALL    CHECK           ;CHECK COMMAND
0578     18 EF                JR      BEGIN0          ;
                     ;
                     ;        READ COMMAND LINE
057A     CD 05CE     RLINE:   CALL    GETC            ;GET CHARACTER
057D     FE 0D                CP      CR              ;CR ?
057F     28 05                JR      Z,RLINE1        ;YES,RLINE1
0581     CD 0255              CALL    DISP            ;DISPLAY IT
0584     18 F4                JR      RLINE           ;
0586     2A F744     RLINE1:  LD      HL,(CURSOR)     ;LOAD CURSOR ADDRESS
0589     DD 2A F746           LD      IX,(RTBLP)      ;LOAD ROW TABLE POINTER
058D     7E          RLINE2:  LD      A,(HL)          ;LOAD SCREEN DATA
058E     CD 02DD              CALL    CONA            ;COVERT TO ASCII
0591     FE 3E                CP      '>'             ;IS THIS PROMPT ?
0593     28 11                JR      Z,RLINE4        ;YES,RLINE4
0595     DD 7E 00             LD      A,(IX+0)        ;LOAD TOP LOWER ADDRESS
0598     E6 FC                AND     0FCH            ;CLEAR FLAGS
059A     BD                   CP      L               ;TOP OF LINE ?
059B     20 06                JR      NZ,RLINE3       ;NO,RLINE3
059D     DD 7E 01             LD      A,(IX+1)        ;LOAD TOP UPPER
05A0     BC                   CP      H               ;TOP OF LINE ?
05A1     28 04                JR      Z,RLINE5        ;YES,RLINE5
05A3     2B          RLINE3:  DEC     HL              ;BACK SCREEN POINTER
05A4     18 E7                JR      RLINE2          ;
05A6     23          RLINE4:  INC     HL              ;SKIP PROMPT
05A7     EB          RLINE5:  EX      DE,HL           ;TOP TO DE
05A8     2A F744              LD      HL,(CURSOR)     ;RELOAD CURSOR ADDRESS
05AB     AF                   XOR     A               ;CLEAR CARRY FOR SBC
05AC     ED 52                SBC     HL,DE           ;GET LINE LENGTH
05AE     44                   LD      B,H             ;LENGTH TO BC
05AF     4D                   LD      C,L             ;
05B0     EB                   EX      DE,HL           ;TOP ADDRESS TO HL
05B1     11 F749              LD      DE,LINEB        ;LOAD LINEB ADDRESS
05B4     38 0E                JR      C,RLINE7        ;JUMP IF ON '>'
05B6     28 0C                JR      Z,RLINE7        ;RLINE7 IF LENGTH=0
05B8     7E          RLINE6:  LD      A,(HL)          ;LOAD SCREEN DATA
05B9     CD 02DD              CALL    CONA            ;CONVERT TO ASCII
05BC     12                   LD      (DE),A          ;MOVE IT TO LINEB
05BD     13                   INC     DE              ;BUMP LINEB POINTER
05BE     23                   INC     HL              ;BUMP SCREEN POINTER
05BF     0B                   DEC     BC              ;COUNT DOWN LENGTH
05C0     78                   LD      A,B             ;ALL MOVED ?
05C1     B1                   OR      C               ;
05C2     20 F4                JR      NZ,RLINE6       ;NO,RLINE6
05C4     3E 0D       RLINE7:  LD      A,CR            ;SET CR IN TAIL
05C6     12                   LD      (DE),A          ;
05C7     CD 0255              CALL    DISP            ;OUTPUT CR
05CA     21 F749              LD      HL,LINEB        ;LOAD LINEB ADDRESS
05CD     C9                   RET                     ;
                                                      ;TO PRINT BUFFER
```

```
                        ;
                        ;       GET CHARACTER
05CE    E5              GETC:   PUSH    HL              ;SAVE HL
05CF    C5                      PUSH    BC              ;SAVE BC
05D0    2A F742         GETC0:  LD      HL,(KBUFOP)     ;LOAD KBUF OUTPUT PTR
05D3    ED 4B F740              LD      BC,(KBUFIP)     ;LOAD KBUF INPUT POINTER
05D7    CD 024F                 CALL    CHLBC           ;BUFFER IS EMPTY ?
05DA    28 F4                   JR      Z,GETC0         ;YES,GETC
05DC    7E                      LD      A,(HL)          ;LOAD DATA
05DD    F5                      PUSH    AF              ;SAVE IT
05DE    23                      INC     HL              ;BUMP KBUFOP
05DF    01 F740                 LD      BC,KBUFE        ;LOAD KBUF END ADDRESS
05E2    CD 024F                 CALL    CHLBC           ;END OF KBUF ?
05E5    20 03                   JR      NZ,GETC1        ;NO,GETC1
05E7    21 F640                 LD      HL,KBUF         ;WRUP AROUND KBUF
05EA    22 F742         GETC1:  LD      (KBUFOP),HL     ;UPDATE KBUFOP
05ED    F1                      POP     AF              ;RESTORE DATA
05EE    C1                      POP     BC              ;RESTORE BC
05EF    E1                      POP     HL              ;RESTORE HL
05F0    C9                      RET                     ;
                        ;       CHECK COMMAND STRING
05F1    21 0000         CHECK:  LD      HL,0            ;PRESET PARAMETERS
05F4    22 F799                 LD      (PAR0),HL       ;
05F7    22 F79B                 LD      (PAR1),HL       ;
05FA    22 F79D                 LD      (PAR2),HL       ;
05FD    21 F749                 LD      HL,LINEB        ;SET LINEB POINTER
0600    46                      LD      B,(HL)          ;LOAD COMAND
0601    23                      INC     HL              ;BUMP BUFFER POINTER
0602    CD 0693                 CALL    CKCMD           ;CHECK COMMAND
0605    20 3B                   JR      NZ,CERR         ;JUMP IF COMMAND ERROR
0607    ED 53 F79F              LD      (RETS),DE       ;SAVE ROUTINE ADDRESS
060B    38 2F                   JR      C,CHECK2        ;JUMP IF CR ONLY
060D    DD 21 F799              LD      IX,PAR0         ;SET PARAM POINTER
0611    06 03                   LD      B,3             ;MAX COUNT OF PARAM
0613    CD 0659         CHECK0: CALL    GETP            ;GET PARAMETER
0616    38 2A                   JR      C,CERR          ;JUMP IF ERROR
0618    DD 73 00                LD      (IX+0),E        ;STORE PARAMETER
061B    DD 72 01                LD      (IX+1),D        ;
061E    DD 23                   INC     IX              ;BUMP PARAM POINTER
0620    DD 23                   INC     IX              ;
0622    FE 0D                   CP      CR              ;CR RELEASE ?
0624    28 02                   JR      Z,CHECK1        ;YES,CHECK1
0626    10 EB                   DJNZ    CHECK0          ;3 PARAMS GOT ?
                                                        ;NO,CHECK0
0628    2A F79F         CHECK1: LD      HL,(RETS)       ;LOAD ROUTINE ADDRESS
062B    E5                      PUSH    HL              ;PUSH IT
062C    CD 03CA                 CALL    HOMEC           ;CURSOR TO HOME
062F    2A F799                 LD      HL,(PAR0)       ;LOAD PARAMETERS
0632    ED 5B F79B              LD      DE,(PAR1)       ;
0636    ED 4B F79D              LD      BC,(PAR2)       ;
063A    AF                      XOR     A               ;INSURE GOOD
063B    C9                      RET                     ;
063C    CD 03CA         CHECK2: CALL    HOMEC           ;CURSOR TO HOME
063F    AF                      XOR     A               ;INSURE GOOD
0640    37                      SCF                     ;SET CR ONLY
0641    C9                      RET                     ;
                        ;
                        ;       COMMAND ERROR
0642    CD 03CA         CERR:   CALL    HOMEC           ;CURSOR TO HOME
0645    CD 04DF                 CALL    MGOUT           ;ERROR MESSAGE
0648    2A 43 4F 4D             DB      '*COMMAND ERROR'
064C    4D 41 4E 44
0650    20 45 52 52
0654    4F 52
0656    0D FF                   DB      CR,0FFH
0658    C9                      RET
```

```
                        ;
                        ;       GET PARAMETER
0659    EB              GETP:   EX      DE,HL           ;BUFFER POINTER TO DE
065A    21 0000                 LD      HL,0            ;INITIALIZE PARAMETER
065D    1A              GETP0:  LD      A,(DE)          ;LOAD DATA
065E    13                      INC     DE              ;BUMP BUFFER POINTER
065F    FE 0D                   CP      CR              ;CR ?
0661    28 0A                   JR      Z,GETP1         ;YES,GETP1
0663    FE 2C                   CP      ','             ;COMMA ?
0665    28 06                   JR      Z,GETP1         ;YES,GETP1
0667    CD 0670                 CALL    GETPX           ;GET PARAM
066A    D8                      RET     C               ;RETURN IF ERROR
066B    18 F0                   JR      GETP0           ;
066D    EB              GETP1:  EX      DE,HL           ;PARAM TO DE
066E    B7                      OR      A               ;INSURE GOOD
066F    C9                      RET                     ;
                        ;
                        ;       GETP SUBROUTINE
0670    CD 0680         GETPX:  CALL    GETH            ;GET HEX VALUE
0673    D8                      RET     C               ;RETURN IF ERROR
0674    C5                      PUSH    BC              ;SAVE BC
0675    4F                      LD      C,A             ;VALUE TO C
0676    06 00                   LD      B,0             ;
0678    29                      ADD     HL,HL           ;SHIFT LEFT BY 4 BITS
0679    29                      ADD     HL,HL           ;
067A    29                      ADD     HL,HL           ;
067B    29                      ADD     HL,HL           ;
067C    09                      ADD     HL,BC           ;INSERT NEW VALUE
067D    C1                      POP     BC              ;RESTORE BC
067E    B7                      OR      A               ;SET NORMAL RETURN
067F    C9                      RET                     ;
                        ;
                        ;       GET HEX VALUE
0680    FE 30           GETH:   CP      '0'             ;LOWER THAN '0' ?
0682    D8                      RET     C               ;YES,RETURN
0683    FE 3A                   CP      '9'+1           ;LARGER THAN 9 ?
0685    38 09                   JR      C,GETH0         ;NO,GETH0
0687    FE 41                   CP      'A'             ;LOWER THAN 'A' ?
0689    D8                      RET     C               ;YES,RETURN
068A    FE 47                   CP      'F'+1           ;LARGER THAN 'F' ?
068C    3F                      CCF                     ;COMPLEMENT CARRY
068D    D8                      RET     C               ;RETURN IF YES
068E    C6 09                   ADD     A,09H           ;ADJUST F0 A-F
0690    E6 0F           GETH0:  AND     0FH             ;CLEAR ZONE
0692    C9                      RET                     ;NORMAL RETURN
                        ;
                        ;       CHECK COMMAND
0693                    CKCMD:
0693    DD 7E 00        CKCMD0: LD      A,(IX+0)        ;LOAD CTBL CONTENT
0696    B8                      CP      B               ;COMMAND MATCH ?
0697    28 0C                   JR      Z,CKCMD1        ;YES,CKCMD1
0699    FE FF                   CP      0FFH            ;TABLE END ?
069B    28 14                   JR      Z,CKCMD2        ;YES,CKCMD2
069D    DD 23                   INC     IX              ;BUMP CTBL POINTER
069F    DD 23                   INC     IX              ;
06A1    DD 23                   INC     IX              ;
06A3    18 EE                   JR      CKCMD0          ;
06A5    DD 5E 01        CKCMD1: LD      E,(IX+1)        ;LOAD ROUTINE ADDRESS
06A8    DD 56 02                LD      D,(IX+2)        ;
06AB    FE 0D                   CP      CR              ;CR ONLY ?
06AD    28 06                   JR      Z,CKCMD3        ;YES,CKCMD3
06AF    AF                      XOR     A               ;INSURE REGAL COMMAND
06B0    C9                      RET                     ;
06B1    3E 01           CKCMD2: LD      A,1             ;CLEAR ZERO FLAG
06B3    B7                      OR      A               ;
06B4    C9                      RET                     ;
```

```
06B5    37              CKCMD3: SCF                          ;SET CR ONLY
06B6    C9                      RET                          ;
                        ;
                        ;       COMMAND TABLE
06B7    4D              CTBL:   DB      'M'                  ;MEMORY EXAMINE & CHANGE
06B8    06E2                    DW      MCMD                 ;
06BA    50                      DB      'P'                  ;PORT EXAMINE & CHANGE
06BB    0797                    DW      PCMD                 ;
06BD    44                      DB      'D'                  ;DISPLAY MEMORY
06BE    07CB                    DW      DCMD                 ;
06C0    41                      DB      'A'                  ;ADD BINARY
06C1    0900                    DW      ACMD                 ;
06C3    53                      DB      'S'                  ;SUBTRACT BINARY
06C4    090A                    DW      SCMD                 ;
06C6    46                      DB      'F'                  ;FILL MEMORY WITH WORD
06C7    090F                    DW      FCMD                 ;
06C9    56                      DB      'V'                  ;MOVE MEMORY
06CA    0923                    DW      VCMD                 ;
06CC    43                      DB      'C'                  ;COMPARE MEMORY
06CD    0932                    DW      CCMD                 ;
06CF    4F                      DB      'O'                  ;SET OFFSET
06D0    0976                    DW      OCMD                 ;
06D2    51                      DB      'Q'                  ;INQUIRE OFFSET
06D3    097A                    DW      QCMD                 ;
06D5    4C                      DB      'L'                  ;LOAD PROGRAM
06D6    0986                    DW      LCMD                 ;
06D8    47                      DB      'G'                  ;GO TO USER PROGRAM
06D9    08F7                    DW      GCMD                 ;
06DB    52                      DB      'R'                  ;PROM UTILITY
06DC    0C00                    DW      RCMD                 ;
06DE    0D                      DB      CR                   ;CARRIAGE RETURN
06DF    0740                    DW      COMRTX               ;NO ACTION
06E1    FF                      DB      0FFH                 ;TABLE SENTINEL
                        ;
                        ;*****************************************************
                        ;****   MEMORY EXAMINE AND CHANGE COMMAND         ****
                        ;*****************************************************
06E2    CD 04F3         MCMD:   CALL    OUTA                 ;DISPLAY '*'
06E5    3E 4D                   LD      A,'M'                ;LOAD 'MEMORY'
06E7    CD 0255                 CALL    DISP                 ;DISPLAY 'L'
06EA    CD 0747                 CALL    OUTW                 ;DISPLAY ADDRESS
06ED    CD 076B                 CALL    OUTS                 ;DISPLAY SPACE
06F0    CD 0770                 CALL    MEMR                 ;GET MEMORY DATA
06F3    CD 074C                 CALL    OUTB                 ;DISPLAY IT
06F6    3E 2D                   LD      A,'-'                ;LOAD '-'
06F8    CD 0255                 CALL    DISP                 ;DISPLAY IT
06FB    CD 05CE                 CALL    GETC                 ;GET DATA FROM KEYBOARD
06FE    FE 0D                   CP      CR                   ;CR ?
0700    28 2C                   JR      Z,MCMD0              ;YES,MCMD0
0702    FE 58                   CP      'X'                  ;EXIT ?
0704    28 34                   JR      Z,MCMD3              ;YES,COMRT
0706    FE 08                   CP      08H                  ;BACK SPACE ?
0708    28 2A                   JR      Z,MCMD2              ;YES,MCMD2
070A    E5                      PUSH    HL                   ;SAVE MEMORY ADDRESS
070B    CD 0785                 CALL    GETPK                ;GET PARAM FROM KEYBOARD
070E    EB                      EX      DE,HL                ;PARAM TO DE
070F    E1                      POP     HL                   ;RESTORE MEMORY ADDRESS
0710    38 2F                   JR      C,COMRT0             ;JUMP IF ERROR FOUND
0712    7B                      LD      A,E                  ;LOAD DATA
0713    CD 077A                 CALL    MEMW                 ;WRITE MEMORY
0716    CD 0770                 CALL    MEMR                 ;RECALL IT
0719    BB                      CP      E                    ;WRITTEN CORRECTLLY ?
071A    28 12                   JR      Z,MCMD0              ;YES,MCMD0
071C    CD 04DF                 CALL    MGOUT                ;'CAN NOT WRITE'
071F    2A 43 41 4E             DB      '*CAN NOT WRITE',0FFH
0723    20 4E 4F 54
```

```
0727    20 57 52 49
072B    54 45 FF
072E    CD 04F8          MCMD0:  CALL    NEWL            ;CHANGE LINE
0731    23               MCMD1:  INC     HL              ;BUMP MEMORY ADDRESS
0732    18 AE                    JR      MCMD            ;
0734    2B               MCMD2:  DEC     HL              ;BACK MEMORY ADDRESS
0735    CD 04F8                  CALL    NEWL            ;NEW LINE
0738    18 A8                    JR      MCMD            ;
073A    CD 0255          MCMD3:  CALL    DISP            ;DISPLAY 'X'
073D    CD 04F8          COMRT:  CALL    NEWL            ;CHANGE LINE
0740    C9               COMRTX: RET                     ;
0741    CD 04F8          COMRT0: CALL    NEWL            ;CHANGE LINE
0744    C3 0642                  JP      CERR            ;
                         ;
                         ;
                         ;       DISPLAY WORD
                         ;       HL=WORD TO BE DISPLAYED
0747    7C               OUTW:   LD      A,H             ;LOAD UPPER BYTE
0748    CD 074C                  CALL    OUTB            ;DISPLAY IT
074B    7D                       LD      A,L             ;LOAD LOWER BYTE
                         ;       FALL THROUGH TO OUTB
                         ;
                         ;       DISPLAY BYTE
                         ;       ACC=BYTE TO BE DISPLAYED
074C    F5               OUTB:   PUSH    AF              ;SAVE BYTE DISPLAYED
074D    07                       RLCA                    ;SHIFT LEFT BY 4 BITS
074E    07                       RLCA                    ;
074F    07                       RLCA                    ;
0750    07                       RLCA                    ;
0751    E6 0F                    AND     0FH             ;CLEAR UPPER HALF
0753    CD 0762                  CALL    CONH            ;CONVERT TO HEX VALUE
0756    CD 0255                  CALL    DISP            ;DISPLAY IT
0759    F1                       POP     AF              ;RESTORE ORIGINAL BYTE
075A    E6 0F                    AND     0FH             ;CLEAR UPPER HALF
075C    CD 0762                  CALL    CONH            ;CONVERT TO HEX VALUE
075F    C3 0255                  JP      DISP            ;DISPLAY A BYTE
                         ;
                         ;       CONVERT TO HEX CHARACTER
0762    FE 0A            CONH:   CP      0AH             ;LOWER THAN 9 ?
0764    38 02                    JR      C,CONH0         ;YES,CONH0
0766    C6 07                    ADD     A,7             ;ADJUSTING F0 A-F
0768    C6 30            CONH0:  ADD     A,30H           ;GET ASCII CODE
076A    C9                       RET                     ;
                         ;
                         ;       DISPLAY A SPACE
076B    3E 20            OUTS:   LD      A,' '           ;LOAD SPACE CODE
076D    C3 0255                  JP      DISP            ;DISPLAY IT
                         ;
                         ;       READ MEMORY DATA WITH OFFSET ADDRESS
                         ;       HL=MEMORY ADDRESS
0770    C5               MEMR:   PUSH    BC              ;SAVE BC
0771    E5                       PUSH    HL              ;SAVE TRUE ADDRESS
0772    ED 4B F7A7               LD      BC,(OFSET)      ;LOAD OFFSET VALUE
0776    09                       ADD     HL,BC           ;GET MEMORY ADDRESS
0777    7E                       LD      A,(HL)          ;LOAD MEMORY DATA
0778    18 08                    JR      MEMW0           ;
                         ;
                         ;       WRITE DATA INTO MEMORY WITH OFFSET
077A    C5               MEMW:   PUSH    BC              ;SAVE BC
077B    E5                       PUSH    HL              ;SAVE TRUE ADDRSS
077C    ED 4B F7A7               LD      BC,(OFSET)      ;LOAD OFFSET VALUE
0780    09                       ADD     HL,BC           ;GET MEMORY ADDRESS
0781    77                       LD      (HL),A          ;WRITE DATA INTO MEMORY
0782    E1               MEMW0:  POP     HL              ;RESTORE TRUE ADDRESS
0783    C1                       POP     BC              ;RESTORE BC
0784    C9                       RET                     ;
```

```
                                ;
                                ;       GET PARAMETER FROM KEYBOARD
0785    21 0000         GETPK:  LD      HL,0            ;CLEAR PARAMETER
0788    CD 0255         GETPKO: CALL    DISP            ;DISPLAY K/B DATA
078B    CD 0670                 CALL    GETPX           ;GET PARAM FROM K/B
078E    D8                      RET     C               ;RETURN IF ERROR
078F    CD 05CE                 CALL    GETC            ;GET DATA FROM KEYBOARD
0792    FE 0D                   CP      CR              ;CR ?
0794    C8                      RET     Z               ;RETURN IF YES
0795    18 F1                   JR      GETPKO          ;
                                ;       PORT EXAMINE AND CHANGE COMMAND
0797    4D              PCMD:   LD      C,L             ;PORT ADDRESS TO C
0798    CD 04F3         PCMDO:  CALL    OUTA            ;DISPLAY '*'
079B    3E 50                   LD      A,'P'           ;SET PORT
079D    CD 0255                 CALL    DISP            ;DISPLAY 'P'
07A0    79                      LD      A,C             ;LOAD PORT ADDRESS
07A1    CD 074C                 CALL    OUTB            ;DISPLAY IT
07A4    CD 076B                 CALL    OUTS            ;PUT SPACE CODE
07A7    ED 78                   IN      A,(C)           ;READ PORT DATA
07A9    CD 074C                 CALL    OUTB            ;DISPLAY IT
07AC    3E 2D                   LD      A,'-'           ;SET '-'
07AE    CD 0255                 CALL    DISP            ;DISPLAY IT
07B1    CD 05CE                 CALL    GETC            ;READ CHARACTER FROM K/B
07B4    FE 0D                   CP      CR              ;CR ?
07B6    CA 07C6                 JP      Z,PCMD1         ;YES,PCMD1
07B9    FE 58                   CP      'X'             ;EXIT ?
07BB    CA 073A                 JP      Z,MCMD3         ;YES,RETURN
07BE    CD 0785                 CALL    GETPK           ;GET PARAM FROM K/B
07C1    DA 0741                 JP      C,COMRTO        ;YES,ERROR RETURN
07C4    ED 69                   OUT     (C),L           ;OUTPUT DATA TO PORT
07C6    CD 04F8         PCMD1:  CALL    NEWL            ;NEW LINE
07C9    18 CD                   JR      PCMDO           ;
                                ;
                                ;       DISPLAY MEMORY COMMAND
07CB    7A              DCMD:   LD      A,D             ;CHECK IF LENGTH=0
07CC    B3                      OR      E               ;ZERO ?
07CD    20 03                   JR      NZ,DCMDO        ;NO,DCMDO
07CF    11 0010                 LD      DE,16           ;ASSUME LENGTH=16
07D2    CD 04F3         DCMDO:  CALL    OUTA            ;DISPLAY '*'
07D5    3E 4D                   LD      A,'M'           ;SET MEMORY
07D7    CD 0255                 CALL    DISP            ;DISPLAY 'M'
07DA    CD 0747                 CALL    OUTW            ;DISPLAY ADDRESS
07DD    D5                      PUSH    DE              ;SAVE LENGTH
07DE    E5                      PUSH    HL              ;SAVE ADDRESS
07DF    0E 00                   LD      C,0             ;SET HEX EXPRESSION
07E1    CD 081C                 CALL    DSUB            ;DISPLAY HEX VALUES
07E4    E1                      POP     HL              ;RESTORE ADDRESS
07E5    D1                      POP     DE              ;RESTORE LENGTH
07E6    06 04                   LD      B,4             ;SET COUNTER FOUR
07E8    CD 076B         DCMD1:  CALL    OUTS            ;OUTPUT SPACE CODE
07EB    10 FB                   DJNZ    DCMD1           ;JUMP IF NOT DONE YET
07ED    0E 01                   LD      C,1             ;SET CHAR. EXPRESSION
07EF    CD 081C                 CALL    DSUB            ;DISPLAY CHARACTERS
07F2    7A                      LD      A,D             ;DONE ALL ?
07F3    B3                      OR      E               ;
07F4    CA 073D                 JP      Z,COMRT         ;YES,RETURN
07F7    CD 0802                 CALL    CBRK            ;CHECK IF BREAK
07FA    C2 073D                 JP      NZ,COMRT        ;RETURN IF BREAK
07FD    CD 04F8                 CALL    NEWL            ;NEW LINE
0800    18 D0                   JR      DCMDO           ;
                                ;
                                ;       CHECK IF BREAK OCCURRED
0802    E5              CBRK:   PUSH    HL              ;SAVE HL
0803    C5                      PUSH    BC              ;SAVE BC
0804    2A F742                 LD      HL,(KBUFOP)     ;LOAD KBUFOP
0807    ED 4B F740              LD      BC,(KBUFIP)     ;LOAD KBUFIP
```

```
080B    CD 024F                 CALL    CHLBC           ;BUFFER IS EMPTY ?
080E    28 09                   JR      Z,CBRK0         ;YES,CBRK0
0810    F3                      DI                      ;DISABLE INTERRUPT
0811    ED 43 F742              LD      (KBUFOP),BC     ;CLEAR BUFFER
0815    FB                      EI                      ;ENABLE INTERRUPT
0816    3E 01                   LD      A,1             ;SET BREAK DETECTED
0818    B7                      OR      A               ;
0819    C1              CBRK0:  POP     BC              ;RESTORE BC
081A    E1                      POP     HL              ;RESTORE HL
081B    C9                      RET                     ;
                        ;
                        ;       DISPAY MEMORY SUBROUTINE
                        ;       C=0     DISPLAY MEMORY DATA IN HEX
                        ;       C=1     DISPLAY MEMORY DATA IN CHARACTER
081C    06 10           DSUB:   LD      B,16            ;SET LINE LENGTH
081E    79              DSUB0:  LD      A,C             ;HEX EXPRESSION ?
081F    B7                      OR      A               ;
0820    20 0B                   JR      NZ,DSUB1        ;NO,DSUB1
0822    CD 076B                 CALL    OUTS            ;OUTPUT SPACE
0825    CD 0770                 CALL    MEMR            ;READ MEMORY DATA
0828    CD 074C                 CALL    OUTB            ;DISPLAY IT IN HEX
082B    18 10                   JR      DSUB2           ;
082D    CD 0770         DSUB1:  CALL    MEMR            ;READ MEMORY DATA
0830    FE 20                   CP      20H             ;WITHIN CHARACTER ?
0832    38 04                   JR      C,DSUB1A        ;NO,DSUB1A
0834    FE 60                   CP      60H             ;WITHIN CHARACTER ?
0836    38 02                   JR      C,DSUB1B        ;YES,DSUB1B
0838    3E 2E           DSUB1A: LD      A,'.'           ;SET PERIOD
083A    CD 0255         DSUB1B: CALL    DISP            ;DISPLAY IT
083D    23              DSUB2:  INC     HL              ;BUMP MEMORY POINTER
083E    1B                      DEC     DE              ;DOWN LENGTH
083F    05                      DEC     B               ;DOWN LINE SIZE
0840    7A                      LD      A,D             ;ALL DONE ?
0841    B3                      OR      E               ;
0842    28 08                   JR      Z,DSUB3         ;YES,DSUB3
0844    78                      LD      A,B             ;ONE LINE END ?
0845    B7                      OR      A               ;
0846    C8                      RET     Z               ;YES,RETURN
0847    7D                      LD      A,L             ;LOAD LOWER ADDRESS
0848    E6 0F                   AND     0FH             ;16 BYTE BOUNDARY ?
084A    20 D2                   JR      NZ,DSUB0        ;NO,DSUB0
084C    78              DSUB3:  LD      A,B             ;END OF LINE ?
084D    B7                      OR      A               ;
084E    C8                      RET     Z               ;YES,RETURN
084F    CD 076B                 CALL    OUTS            ;OUTPUT SPACE
0852    79                      LD      A,C             ;LOAD EXPRESSION PARAM
0853    B7                      OR      A               ;HEX EXPRESSION ?
0854    20 06                   JR      NZ,DSUB4        ;NO,DSUB4
0856    CD 076B                 CALL    OUTS            ;OUTPUT TWO SPACES
0859    CD 076B                 CALL    OUTS            ;
085C    05              DSUB4:  DEC     B               ;DOWN LINE SIZE
085D    18 ED                   JR      DSUB3           ;
                        ;
                        ;       DISPLAY REGISTER ROUTINE
085F    22 F7A1         RDISP:  LD      (HLSAV),HL      ;SAVE HL
0862    ED 73 F7A3              LD      (SPSAV),SP      ;SAVE SP
0866    2A F7A3                 LD      HL,(SPSAV)      ;LOAD OLD STACK POINTER
0869    E5                      PUSH    HL              ;STACK IT
086A    DD E5                   PUSH    IX              ;STACK IX
086C    FD E5                   PUSH    IY              ;STACK IY
086E    F5                      PUSH    AF              ;STACK PSW
086F    C5                      PUSH    BC              ;STACK BC
0870    D5                      PUSH    DE              ;STACK DE
0871    2A F7A1                 LD      HL,(HLSAV)      ;LOAD HL VALUE
0874    E5                      PUSH    HL              ;STACK HL
0875    08                      EX      AF,AF'          ;FLIP TO THE OTHER STACK
```

```
0876    D9                      EXX                     ;
0877    F5                      PUSH    AF              ;STACK PSW'
0878    C5                      PUSH    BC              ;STACK BC'
0879    D5                      PUSH    DE              ;STACK DE'
087A    E5                      PUSH    HL              ;STACK HL'
087B    08                      EX      AF,AF'          ;RESTORE REG. STACK
087C    D9                      EXX                     ;
                        ;       SAVED ALL REGISTERS
087D    2A F7A3                 LD      HL,(SPSAV)      ;LOAD OLD STACK POINTER
0880    23                      INC     HL              ;ADJUSTING
0881    06 00                   LD      B,0             ;SET 16 BITS REGISTER
0883    11 08CB                 LD      DE,RMSG         ;SET RMSG POINTER
0886    CD 04F8         RDSP0:  CALL    NEWL            ;NEW LINE
0889    CD 04F3                 CALL    OUTA            ;DISPLAY '*'
088C    1A              RDSP1:  LD      A,(DE)          ;LOAD RMSG CONTENT
088D    13                      INC     DE              ;BUMP RMSG POINTER
088E    FE FF                   CP      0FFH            ;END OF LINE ?
0890    20 08                   JR      NZ,RDSP2        ;NO,RDSP2
0892    1A                      LD      A,(DE)          ;LOAD RMSG CONTENT
0893    FE FF                   CP      0FFH            ;END OF DISPLAY ?
0895    28 26                   JR      Z,RDSP4         ;YES,RDSP4
0897    04                      INC     B               ;SET 8 BITS REGISTER
0898    18 EC                   JR      RDSP0           ;
089A    CD 0255         RDSP2:  CALL    DISP            ;DISPLAY REG. NAME
089D    1A                      LD      A,(DE)          ;LOAD REG. NAME
089E    CD 0255                 CALL    DISP            ;DISPLAY IT
08A1    13                      INC     DE              ;BUMP RMSG POINTER
08A2    3E 3D                   LD      A,'='           ;LOAD '='
08A4    CD 0255                 CALL    DISP            ;DISPLAY IT
08A7    7E                      LD      A,(HL)          ;LOAD STACK CONTENTS
08A8    2B                      DEC     HL              ;PUSH DOWN STACK
08A9    CD 074C                 CALL    OUTB            ;DISPLAY REG. VALUE
08AC    78                      LD      A,B             ;16 BITS REGISTER ?
08AD    B7                      OR      A               ;
08AE    20 05                   JR      NZ,RDSP3        ;NO,RDSP3
08B0    7E                      LD      A,(HL)          ;LOAD STACK CONTENT
08B1    CD 074C                 CALL    OUTB            ;DISPLAY IT
08B4    2B                      DEC     HL              ;PUSH DOWN STACK
08B5    CD 076B         RDSP3:  CALL    OUTS            ;OUTPUT TWO SPACES
08B8    CD 076B                 CALL    OUTS            ;
08BB    18 CF                   JR      RDSP1           ;
                        ;       DISPLAYED ALL REGISTERS
08BD    CD 04F8         RDSP4:  CALL    NEWL            ;NEW LINE
08C0    31 0000                 LD      SP,STACK        ;RESET STACK POINTER
08C3    21 0569                 LD      HL,BEGIN0       ;SET RETURN PASS
08C6    E5                      PUSH    HL              ;
08C7    D3 19                   OUT     (CNMI),A        ;CLEAR NMI FF
08C9    ED 45                   RETN                    ;
                        ;
08CB    50 43 53 50     RMSG:   DB      'PCSPIXIY',0FFH
08CF    49 58 49 59
08D3    FF
08D4    41 20 46 20             DB      'A F B C D E H L ',0FFH
08D8    42 20 43 20
08DC    44 20 45 20
08E0    48 20 4C 20
08E4    FF
08E5    41 22 46 22             DB      'A"F"B"C"D"E"H"L"',0FFH,0FFH
08E9    42 22 43 22
08ED    44 22 45 22
08F1    48 22 4C 22
08F5    FF FF
                        ;
                        ;       GO COMMAND
08F7    ED 4B F7A7      GCMD:   LD      BC,(OFSET)      ;LOAD OFFSET VALUE
08FB    09                      ADD     HL,BC           ;GET TRUE ADDRESS
```

```
08FC    31 0000              LD     SP,STACK       ;SET MONITOR STACK
08FF    E9                   JP     (HL)           ;GO ! GO ! GO !
                     ;
                     ;       ADD BINARY COMMAND
0900    19           ACMD:   ADD    HL,DE          ;DO ADDITION
0901    CD 04F3      ACMD0:  CALL   OUTA           ;DISPLAY '*'
0904    CD 0747              CALL   OUTW           ;DISPLAY RESULT
0907    C3 073D              JP     COMRT          ;RETURN
                     ;
                     ;       SUBTRACT BINARY COMMAND
090A    AF           SCMD:   XOR    A              ;CLEAR CARRY
090B    ED 52                SBC    HL,DE          ;DO SUBTRACT
090D    18 F2                JR     ACMD0          ;
                     ;
                     ;       FILL MEMORY WITH WORD COMMAND
                     ;       HL=MEMORY ADDRESS
                     ;       DE=LENGTH
                     ;       BC=DATA TO BE FILLED
090F    7A           FCMD:   LD     A,D            ;COMMAND DONE ?
0910    B3                   OR     E              ;
0911    C8                   RET    Z              ;YES,RETURN
0912    78                   LD     A,B            ;LOAD UPPER DATA
0913    CD 077A              CALL   MEMW           ;WRITE IT INTO MEMORY
0916    1B                   DEC    DE             ;DOWN LENGTH
0917    7A                   LD     A,D            ;COMMAND DONE ?
0918    B3                   OR     E              ;
0919    C8                   RET    Z              ;YES,RETURN
091A    23                   INC    HL             ;BUMP MEMORY ADDRESS
091B    79                   LD     A,C            ;LOAD LOWER DATA
091C    CD 077A              CALL   MEMW           ;WRITE IT INTO MEMORY
091F    23                   INC    HL             ;BUMP MEMORY ADDRESS
0920    1B                   DEC    DE             ;DOWN LENGTH
0921    18 EC                JR     FCMD           ;
                     ;
                     ;       MOVE MEMORY COMMAND
                     ;       HL=DESTINATION ADDRESS
                     ;       DE=SOURCE ADDRESS
                     ;       BC=LENGTH
0923    78           VCMD:   LD     A,B            ;COMMAND DONE ?
0924    B1                   OR     C              ;
0925    C8                   RET    Z              ;YES,RETURN
0926    C5                   PUSH   BC             ;SAVE LENGTH
0927    ED 4B F7A7           LD     BC,(OFSET)     ;LOAD OFFSET VALUE
092B    09                   ADD    HL,BC          ;GET TRUE DESTINATION
092C    EB                   EX     DE,HL          ;SOURCE ADDRESS TO HL
092D    09                   ADD    HL,BC          ;GET TRUE SOURCE
092E    C1                   POP    BC             ;RESTORE LENGTH
092F    ED B0                LDIR                  ;
0931    C9                   RET                   ;COMMAND END
                     ;
                     ;       COMPARE MEMORY COMMAND
0932    78           CCMD:   LD     A,B            ;COMMAND DONE ?
0933    B1                   OR     C              ;
0934    C8                   RET    Z              ;YES,RETURN
0935    C5           CCMD0:  PUSH   BC             ;SAVE LENGTH
0936    CD 0770              CALL   MEMR           ;READ MEMORY (OP1)
0939    47                   LD     B,A            ;SAVE IT
093A    EB                   EX     DE,HL          ;OP2 ADDRESS TO HL
093B    CD 0770              CALL   MEMR           ;READ MEMORY (OP2)
093E    EB                   EX     DE,HL          ;RESTORE ADDRESS
093F    4F                   LD     C,A            ;SAVE OP2
0940    B8                   CP     B              ;DATA MATCH ?
0941    20 06                JR     NZ,CCMD2       ;NO,CCMD2
0943    C1                   POP    BC             ;RESTORE LENGTH
0944    23           CCMD1:  INC    HL             ;BUMP OP1 POINTER
0945    13                   INC    DE             ;BUMP OP2 POINTER
```

```
0946    0B              DEC     BC              ;DOWN LENGTH
0947    18 E9           JR      CCMD            ;
0949    CD 04F3  CCMD2: CALL    OUTA            ;DISPLAY '*'
094C    CD 0747         CALL    OUTW            ;DISPLAY OP1 ADDRESS
094F    3E 3A           LD      A,':'           ;LOAD ':'
0951    CD 0255         CALL    DISP            ;DISPLAY IT
0954    78              LD      A,B             ;LOAD OP1 DATA
0955    CD 074C         CALL    OUTB            ;DISPLAY IT
0958    CD 076B         CALL    OUTS            ;OUTPUT TWO SPACES
095B    CD 076B         CALL    OUTS            ;
095E    EB              EX      DE,HL           ;OP2 ADDRESS TO HL
095F    CD 0747         CALL    OUTW            ;DISPLAY OP2 ADDRESS
0962    EB              EX      DE,HL           ;RESTORE ADDRESS
0963    3E 3A           LD      A,':'           ;LOAD ':'
0965    CD 0255         CALL    DISP            ;DISPLAY IT
0968    79              LD      A,C             ;LOAD OP2 DATA
0969    CD 074C         CALL    OUTB            ;DISPLAY IT
096C    CD 04F8         CALL    NEWL            ;NEW LINE
096F    CD 0802         CALL    CBRK            ;CHECK IF BREAK
0972    C1              POP     BC              ;RESTORE LENGTH
0973    C0              RET     NZ              ;RETURN IF BREAK
0974    18 CE           JR      CCMD1           ;
                 ;
                 ;      SET OFFSET COMMAND
0976    22 F7A7  OCMD:  LD      (OFSET),HL      ;STORE OFFSET
0979    C9              RET                     ;
                 ;
                 ;      INQUIRE OFFSET COMMAND
097A    CD 04F3  QCMD:  CALL    OUTA            ;DISPLAY '*'
097D    2A F7A7         LD      HL,(OFSET)      ;LOAD OFFSET VALUE
0980    CD 0747         CALL    OUTW            ;DISPLAY IT
0983    C3 073D         JP      COMRT           ;COMMAND END
                 ;
                 ;      LOAD PROGRAM COMMAND
0986    CD 0A43  LCMD:  CALL    ISIO            ;INITIALIZE SIO
0989    1E 00           LD      E,0             ;RESET ERROR FLAG
098B    CD 0A36  LCMD0: CALL    DIN             ;READ DATA FROM SIO
098E    FE 3A           CP      ':'             ;START OF RECORD ?
0990    20 F9           JR      NZ,LCMD0        ;NO,LCMD0
0992    0E 00           LD      C,0             ;RESET CHECK SUM
0994    CD 0A1C         CALL    GETBS           ;GET BYTE FROM SIO
0997    30 04           JR      NC,LCMD2        ;JUMP IF NOT ERROR
0999    CB C3    LCMD1: SET     0,E             ;SET INVALID CHARACTER
099B    18 EE           JR      LCMD0           ;
099D    47       LCMD2: LD      B,A             ;SAVE TEXT LENGTH
                 ;      B=TEXT LENGTH
                 ;      C=CHECK SUM
                 ;      E=ERROR CODE
099E    CD 0A1C         CALL    GETBS           ;GET BYTE FROM SIO
09A1    38 F6           JR      C,LCMD1         ;JUMP IF INVALID CHAR.
09A3    67              LD      H,A             ;SAVE UPPER ADDRESS
09A4    CD 0A1C         CALL    GETBS           ;GET BYTE FROM SIO
09A7    38 F0           JR      C,LCMD1         ;JUMP IF INVALID CHAR.
09A9    6F              LD      L,A             ;SAVE LOWER ADDRESS
09AA    CD 0A1C         CALL    GETBS           ;GET BYTE FROM SIO
09AD    38 EA           JR      C,LCMD1         ;JUMP IF INVALID CHAR.
09AF    FE 01           CP      01H             ;END RECORD ?
09B1    28 18           JR      Z,LCMD4         ;YES,LCMD4
09B3    CD 0A1C  LCMD3: CALL    GETBS           ;GET BYTE FROM SIO
09B6    38 E1           JR      C,LCMD1         ;JUMP IF INVALID CHAR.
09B8    CD 077A         CALL    MEMW            ;WRITE IT INTO MEMORY
09BB    23              INC     HL              ;BUMP MEMORY ADDRESS
09BC    10 F5           DJNZ    LCMD3           ;ONE RECORD DONE  ?
                                                ;NO,LCMD3
09BE    CD 0A1C         CALL    GETBS           ;GET BYTE FROM SIO
09C1    38 D6           JR      C,LCMD1         ;JUMP IF INVALID CHAR.
```

```
09C3    79                      LD      A,C             ;LOAD CHACK SUM
09C4    B7                      OR      A               ;SUM GOOD ?
09C5    28 C4                   JR      Z,LCMD0         ;YES,LCMD0
09C7    CB CB                   SET     1,E             ;SET SUM ERROR
09C9    18 C0                   JR      LCMD0           ;
09CB    CD 0A1C         LCMD4:  CALL    GETBS           ;READ CHECK SUM
09CE    38 C9                   JR      C,LCMD1         ;JUMP IF INVALID CHAR.
09D0    79                      LD      A,C             ;LOAD SUM
09D1    B7                      OR      A               ;SUM GOOD ?
09D2    28 02                   JR      Z,LCMD5         ;YES,LCMD5
09D4    CB CB                   SET     1,E             ;SET CHECK SUM ERROR
09D6    CD 0A36         LCMD5:  CALL    DIN             ;IGNORE CR/LF
09D9    CD 0A36                 CALL    DIN             ;
09DC    CB 43                   BIT     0,E             ;INVALID CHARACTER ?
09DE    28 20                   JR      Z,LCMD6         ;NO,LCMD6
09E0    CD 04DF                 CALL    MGOUT           ;INVALID CHARACTER
09E3    2A 49 4E 56             DB      '*INVALID CHARACTER DETECTED',CR,0FFH
09E7    41 4C 49 44
09EB    20 43 48 41
09EF    52 41 43 54
09F3    45 52 20 44
09F7    45 54 45 43
09FB    54 45 44 0D
09FF    FF
0A00    CB 4B           LCMD6:  BIT     1,E             ;CHECK SUM ERROR ?
0A02    28 15                   JR      Z,LCMD7         ;NO,LCMD7
0A04    CD 04DF                 CALL    MGOUT           ;SUM ERROR
0A07    2A 43 48 45             DB      '*CHECK SUM ERROR',CR,0FFH
0A0B    43 4B 20 53
0A0F    55 4D 20 45
0A13    52 52 4F 52
0A17    0D FF
0A19    C3 073D         LCMD7:  JP      COMRT           ;COMMAND END
                        ;
                        ;       GET BYTE FROM SIO
0A1C    CD 0A36         GETBS:  CALL    DIN             ;READ SIO
0A1F    CD 0680                 CALL    GETH            ;GET HEX VALUE
0A22    D8                      RET     C               ;RETURN IF ERROR
0A23    07                      RLCA                    ;SHIFT LEFT BY 4 BITS
0A24    07                      RLCA                    ;
0A25    07                      RLCA                    ;
0A26    07                      RLCA                    ;
0A27    57                      LD      D,A             ;SAVE HEX DIGIT
0A28    CD 0A36                 CALL    DIN             ;READ DATA FROM SIO
0A2B    CD 0680                 CALL    GETH            ;GET HEX VALUE
0A2E    D8                      RET     C               ;RETURN IF ERROR
0A2F    B2                      OR      D               ;CONSTRUCT BYTE DATA
0A30    57                      LD      D,A             ;SAVE IT
0A31    81                      ADD     A,C             ;ACCUMULATE SUM
0A32    4F                      LD      C,A             ;UPDATE SUM
0A33    7A                      LD      A,D             ;LOAD DATA
0A34    B7                      OR      A               ;SET NO ERROR
0A35    C9                      RET                     ;
                        ;
                        ;       READ DATA FROM SIO
0A36    DB 01           DIN:    IN      A,(SIOAC)       ;READ RR0
0A38    E6 01                   AND     01H             ;RECEIVER READY ?
0A3A    28 FA                   JR      Z,DIN           ;NO,DIN
0A3C    DB 00                   IN      A,(SIOAD)       ;READ DATA FROM SIO
0A3E    E6 7F                   AND     7FH             ;CLEAR MSB
0A40    C3 0255                 JP      DISP            ;
                        ;
                        ;       INITIALIZE SIO CHANNEL A
0A43    3E 03           ISIO:   LD      A,03H           ;RESET SIO
0A45    D3 08                   OUT     (CTC0),A        ;
0A47    3E 45                   LD      A,45H           ;SET COUNTER MODE
```

```
0A49    D3 08               OUT     (CTCO),A          ;
0A4B    3E 18               LD      A,24              ;LOAD TIME CONSTANT
0A4D    D3 08               OUT     (CTCO),A          ;
0A4F    21 0A5F             LD      HL,SIOACC         ;LOAD COMMAND ADDRESS
0A52    0E 01               LD      C,SIOAC           ;LOAD SIOAC ADDRESS
0A54    06 0A               LD      B,SIOACL-SIOACC   ;LOAD COMMAND LENGTH
0A56    ED B3               OTIR                      ;INITIALIZE SIO CHNL A
0A58    DB 00               IN      A,(SIOAD)         ;DUMMY READ
0A5A    DB 00               IN      A,(SIOAD)         ;
0A5C    DB 00               IN      A,(SIOAD)         ;
0A5E    C9                  RET                       ;
                    ;       SIO CHANNEL A COMMAND CHAIN
0A5F    18          SIOACC: DB      18H               ;RESET CHANNEL
0A60    04                  DB      04H               ;SELECT REG. 4
0A61    CC                  DB      0CCH              ;X64 CLOCK, TWO STOP BIT
0A62    05                  DB      05H               ;SELECT REG. 5
0A63    62                  DB      62H               ;8BITS/CHAR.,RTS ON
                                                      ;RTS ON
0A64    01                  DB      01H               ;SELECT REG. 1
0A65    00                  DB      00H               ;DISABLE INTERRUPT
0A66    03                  DB      03H               ;SELECT REG. 3
0A67    C1                  DB      0C1H              ;8 BITS/CHAR., RX ENABLE
0A68    6A                  DB      6AH               ;ENABLE XMITTER
0A69                SIOACL  EQU     $
                    ;
                    ;       NMI ROUTINE
0A69    F5          NMI:    PUSH    AF                ;SAVE PSW
0A6A    C5                  PUSH    BC                ;SAVE BC
0A6B    D5                  PUSH    DE                ;SAVE DE
0A6C    E5                  PUSH    HL                ;SAVE   HL
0A6D    CD 04DF             CALL    MGOUT             ;*NMI DETECTED
0A70    0D 2A 4E 4D         DB      CR,'*NMI DETECTED',0FFH
0A74    49 20 44 45
0A78    54 45 43 54
0A7C    45 44 FF
0A7F    E1                  POP     HL                ;RESTORE HL
0A80    D1                  POP     DE                ;RESTORE DE
0A81    C1                  POP     BC                ;RESTORE BC
0A82    F1                  POP     AF                ;RESTORE PSW
0A83    C3 085F             JP      RDISP             ;DISPLAY REGISTERS
                    ;
                    ;
                    ;
                    ;       RST 08H ROUTINE
0A86    22 F7A9     RST08:  LD      (HLSAVO),HL       ;SAVE HL
0A89    ED 43 F7AB          LD      (BCSAVO),BC       ;SAVE BC
0A8D    E3                  EX      (SP),HL           ;GET PC VALUE
0A8E    4E                  LD      C,(HL)            ;LOAD VECOR
0A8F    06 00               LD      B,0               ;CLEAR B
0A91    23                  INC     HL                ;HL POINTS RETURN ADDRES
0A92    E3                  EX      (SP),HL           ;STACK RETURN ADDRESS
0A93    21 0AA3             LD      HL,RST08T         ;LOAD JUMP TABLE ADDRESS
0A96    09                  ADD     HL,BC             ;POINTS TO ROUTINE TABLE
0A97    4E                  LD      C,(HL)            ;LOAD LOWER ADDRESS
0A98    23                  INC     HL                ;BUMP TABLE POINTER
0A99    46                  LD      B,(HL)            ;LOAD UPPER ADDRESS
0A9A    C5                  PUSH    BC                ;STACK JUMP ADDRESS
0A9B    ED 4B F7AB          LD      BC,(BCSAVO)       ;RESTORE BC
0A9F    2A F7A9             LD      HL,(HLSAVO)       ;RESTORE HL
0AA2    C9                  RET                       ;GOTO EACH ROUTINE
                    ;
                    ;       JUMP TABLE
0AA3    0255        RST08T: DW      DISP              ;00
0AA5    04DF                DW      MGOUT             ;02
0AA7    057A                DW      RLINE             ;04
0AA9    05CE                DW      GETC              ;06
```

```
0AAB    0659              DW      GETP             ;08
0AAD    0670              DW      GETPX            ;10
0AAF    05F1              DW      CHECK            ;12
0AB1    0693              DW      CKCMD            ;14
0AB3    0785              DW      GETPK            ;16
0AB5    074C              DW      OUTB             ;18
0AB7    0747              DW      OUTW             ;20
0AB9    0680              DW      GETH             ;22
0ABB    0802              DW      CBRK             ;24
0ABD    02E3              DW      QPDT             ;26
0ABF    0ACD              DW      PRST             ;28
0AC1    0000              DW      0                ;30
0AC3    0000              DW      0                ;32
0AC5    0000              DW      0                ;34
0AC7    0000              DW      0                ;36
0AC9    0000              DW      0                ;38
0ACB    0000              DW      0                ;40
                  ;
                  ;       GET PRINTER STATUS
0ACD    3A F7B1   PRST:   LD      A,(PFLAG)        ;LOAD PRINTER FLAG
0AD0    B7                OR      A                ;SET CC
0AD1    C9                RET                      ;
                  ;
                  ;
                  ;
                          ORG     START+0C00H
0C00              RCMD    EQU     $                ;ROM COMMAND ADDRESS
                  ;
                  ;
                          END
```

Appendix 2 Z80周辺LSIの割り込み制御について

Z80周辺LSIはZ80 CPUのモード2の割り込みを可能にするロジックを内蔵しています．したがって，他のプロセッサ・ファミリにおいてみられるような，割り込み制御専用のLSIはZ80ファミリにはありません．Z80周辺LSIの割り込み制御の特徴はデイジー・チェイン方式（daisy chain：ひなぎくの花輪，花づなの意）にあります．またこれらのLSIはCPUがRETI命令を実行するのを監視するロジックを持っており，同命令が実行されると割り込み要求をクリヤする機能を持っています．

デイジー・チェイン方式

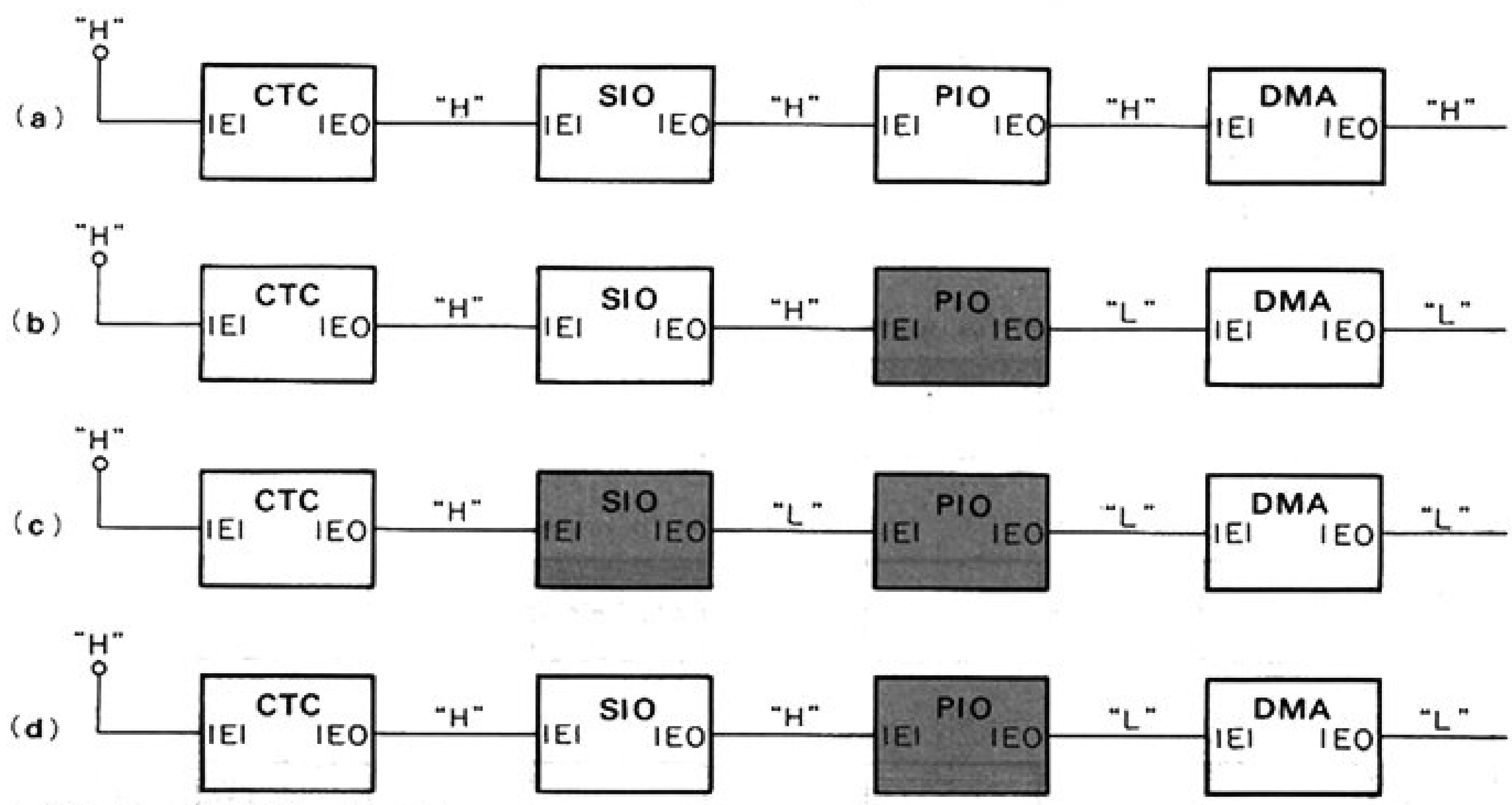
Z80周辺LSIの割り込み優先順位制御は，デイジー・チェイン方式により行われます．図1に四つの周辺LSI間のデイジー・チェインの例を示します．この図からわかるように，隣接したチップのIEO(Interrupt Enable Output)端子とIEI(Interrupt Enable Input)端子は相互に接続され，最高優先順位を持つチップ（左端のCTC）のIEI端子は常に"H"レベルになっています．この図では左端のCTCが最高優先順位を持ち，右へ行くほど順位が下がり，右端のDMAは最低位の順位を持っています．

各チップは，自分のIEI端子が"H"レベルにあるときにのみ割り込み要求（$\overline{INT}$）を出すことができ，割り込み要求を出すと自分のIEOを"L"レベルにします．また自分のIEI端子が"L"レベルであるときにはどのような場合でも，IEO端子を"L"レベルにします．

図1(a)はすべてのチップが割り込み要求を持っていない状態であり，IEIおよびIEOはすべて"H"レベルになっています．

図(b)はPIOだけが割り込み要求を行っている状態であり，PIOは自分のIEO端子を"L"レベルにし，DMAの割り込み要求を抑えています．CPUにより割り込みが受け付けられ，かつプログラムによりRETI命令が実行されると，PIOは割り込み要求をクリヤし，IEOを"H"レベルにします．

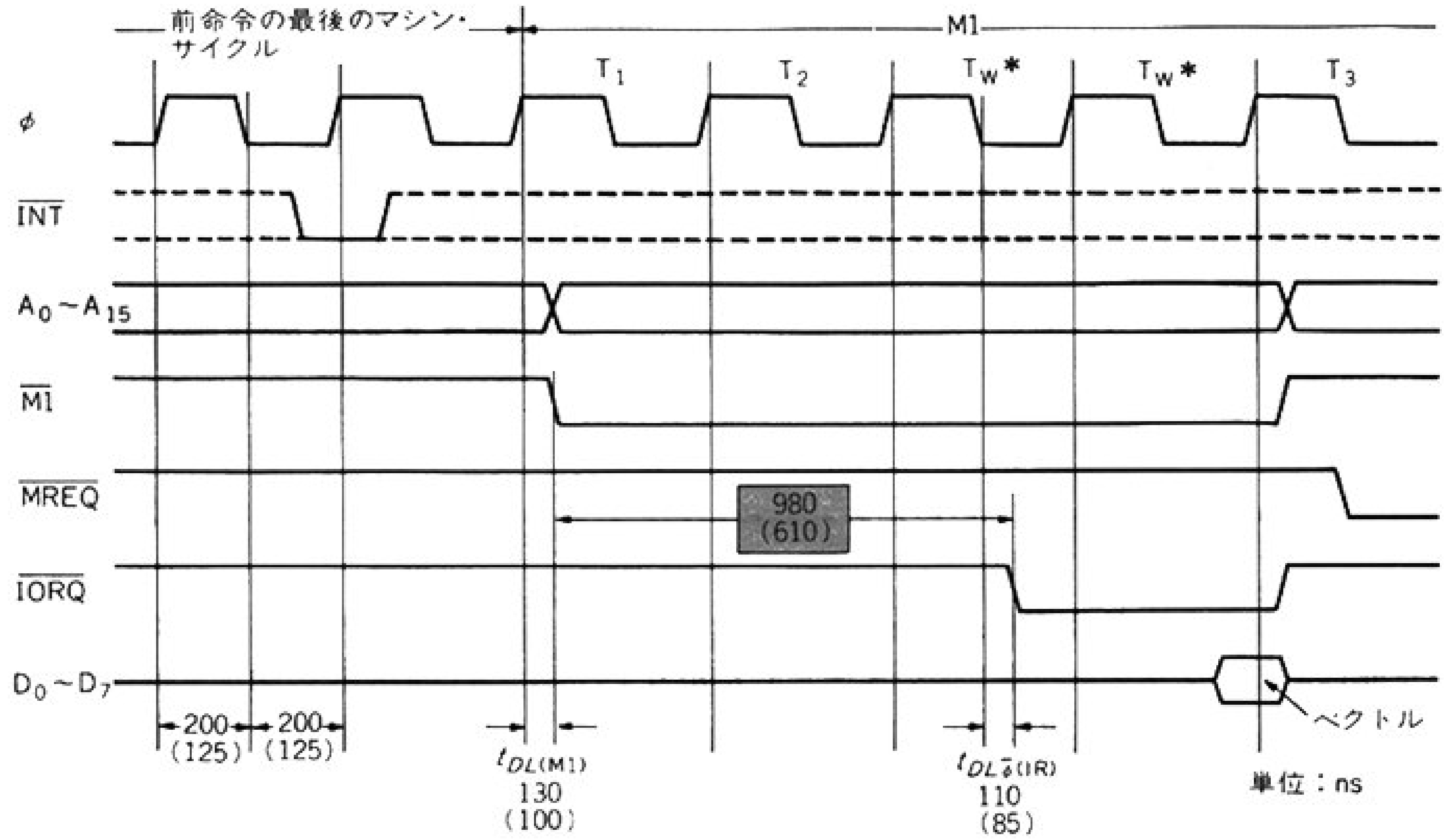
図(c)はSIOとPIOが割り込み要求を持っているが，優先順位の高いSIOだけが割り込み要求を行っている状態です．SIOは自分のIEO端子を"L"



(a)割り込み要求を持っているものがない状態 (c)SIOとPIOが割り込み要求を持っている
(b)PIOが割り込み要求を持っている (d)SIOの割り込みがクリアされ，PIOが割り込み要求を行っている

〈図1〉
割り込みデイジー・チェインの例

〈図2〉
Z80-CPUの INTA サイクルのタイミング



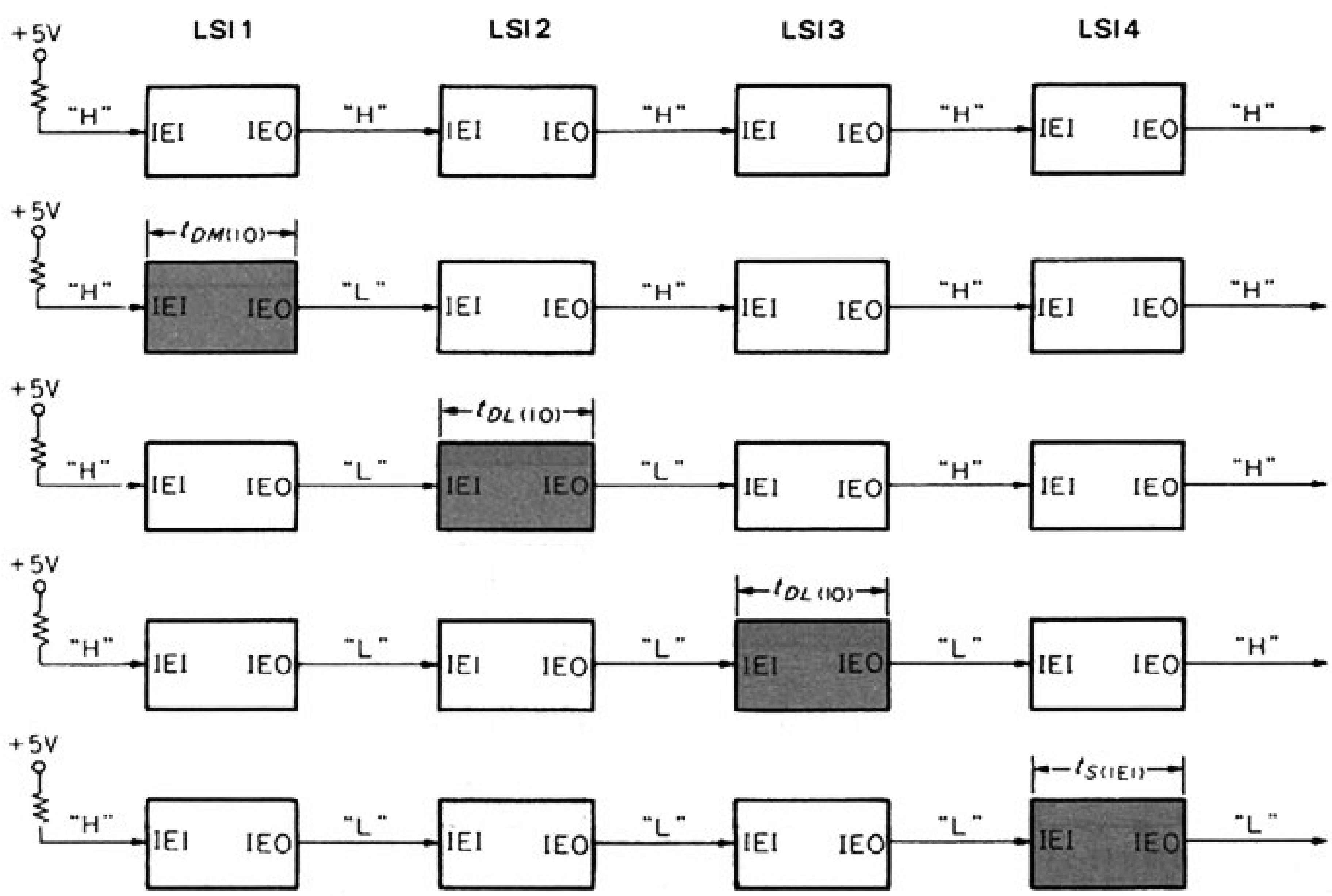
M1の立ち下がりから$\overline{IORQ}$が立ち下がるまでにIEIおよびIEOは伝搬する必要がある．
・カッコ内はZ80A-CPUの値
・φのデューティ・サイクルは50%であるとする．

レベルにしてＰＩＯの割り込み要求を抑えています．ＳＩＯの割り込みが受け付けられ，プログラムにより RETI命令が実行されると，ＳＩＯの割り込み要求はクリヤされ，そのＩＥＯは"H"レベルになります〔図(d)〕．これによりそれまで割り込み要求を出すことを抑えられていたＰＩＯは，ＣＰＵに対して割り込み要求を出すことができます．ＰＩＯの割り込みがクリヤされると，図(a)の状態に戻ります．

デイジー・チェインの拡張

デイジー・チェイン状に接続されたＩＥＩおよびＩＥＯ信号は，あたかもカスケード接続されたアダー間のリプル・キャリのように，チップ間を伝搬します．したがってこの伝搬時間を考えると，デイジー・チェイン可能なチップ数にはある限度があると想像するのはごく容易です．この想像は誤りではなく，事実限度があるのです．ではその限度とはいくつであるのか考えてみましょう．

〈図3〉
$\overline{M1}$の直前にＬＳＩ１が割り込み要求を発生した場合のIEI/IEOの伝搬のようす



〈表1〉 Z80/Z8Aファミリの周辺LSIのIEI，IEOに関するタイミング規定

LSI			PIO		CTC		SIO/DART		DMA	
関連信号	記号	説明	Z80	Z80A	Z80	Z80A	Z80	Z80A	Z80	Z80A
IEI	$t_{S(IEI)}$	INTAサイクル中の$\overline{\text{IORQ}}$の立ち下がり時に対するIEIセットアップ・タイム	140	140	200	140	200	140	140	140
IEO	$t_{DH(IO)}$	IEIの立ち上がり時からのIEO遅延時間	210	160	220	160	150	100	210	160
	$t_{DL(IO)}$	IEIの立ち下がり時からのIEO遅延時間	190	130	190	130	150	100	190	130
	$t_{DM(IO)}$	$\overline{\text{M1}}$の立ち下がり時からのIEO遅延時間（$\overline{\text{M1}}$の直前に割り込みを生じたとき）	300	190	300	190	300	190	300	190

まずZ80 CPUの割り込み認識シーケンス(INTAサイクル)を復習してみます(図2)．Z80 CPUは命令の最後のマシン・サイクルにおいて割り込み要求($\overline{\text{INT}}$)を検出し，次のマシン・サイクルにINTAサイクルに入ります．このマシン・サイクルではまず$\overline{\text{M1}}$を出力し(T_1)，その後$\overline{\text{IORQ}}$を出力します($T_w{}^*$)．またこのサイクルには自動的に二つの$T_w{}^*$ステートが挿入され，IEIおよびIEOが周辺LSI間のデイジー・チェインを伝搬する時間を補償します．

次に，このINTAサイクルにおける周辺LSIのふるまいを考えてみます．Z80ファミリの周辺LSIは$\overline{\text{M1}}$が“L”レベルである期間，割り込み要求状態が変化しないように設計されており，$\overline{\text{IORQ}}$が出された時点にIEIが“H”レベルである周辺LSIだけが割り込みベクトルをバスに乗せます．したがって，$\overline{\text{M1}}$が出てから$\overline{\text{IORQ}}$が出るまでの間に，IEIおよびIEOはデイジー・チェインの終端まで伝搬する必要があります．

IEIおよびIEOがデイジー・チェインを伝搬する時間の最悪値は，最高優先順位を持つ周辺LSIの割り込み要求が$\overline{\text{M1}}$の直前に発生したときに起こります．この場合のIEIおよびIEOの伝搬状態を図3に示します．図中の$t_{DM(IO)}$は$\overline{\text{M1}}$の直前に割り込みが発生したときの，$\overline{\text{M1}}$の立ち下がり時点からのIEO遅延時間です．$t_{DL(IO)}$はIEIの立ち下がり時点からのIEOの遅延時間です．また$t_{S(IEI)}$はINTA時の$\overline{\text{IORQ}}$の立ち下がりに対するIEIセットアップ時間です．

したがって，図3のデイジー・チェインをIEIおよびIEOが伝搬する時間t_Pは，

$$t_P = t_{DM(IO)} + t_{DL(IO)} + t_{DL(IO)} + t_{S(IEI)} \cdots\cdots\cdots\cdots\cdots (1)$$

となります．そしてこの時間は図2に示した$\overline{\text{M1}}$の立ち下がりから$\overline{\text{IORQ}}$の立ち下がりまでの時間(Z80：980ns，Z80A：610ns)より小さい必要があります．

$t_{DM(IO)}$，$t_{DL(IO)}$，$t_{S(IEI)}$などの値は各LSIにより異なります．表1に，Z80およびZ80Aファミリの周辺LSIに関するこれらの値の最大値を示します．では，実際にはいくつの周辺LSIをデイジー・チェインできるのかを考えてみましょう．

簡単のため図3のLSI1～LSI4として，4個のPIOを用いることにします．表1から各定数を式(1)に代入すると，

・**Z80ファミリの場合**：

$$t_{P(Z80)} = 300+190+190+140 = 820〔ns〕< 980〔ns〕$$

また，

・**Z80Aファミリの場合**：

$$t_{P(Z80A)} = 190+130+130+140 = 590〔ns〕< 610〔ns〕$$

となります．

したがって，図3のLSI1～LSI4として，PIO4個をデイジー・チェインすることは可能です．

ではPIO5個をデイジー・チェインさせた場合はどうでしょう．

この場合には，

$$t_{P(Z80)} = 300+190+190+190+140 = 1010〔ns〕> 980〔ns〕$$

$$t_{P(Z80A)} = 190+130+130+130+140 = 720〔ns〕> 610〔ns〕$$

となり，図2に示した条件を満足しません．

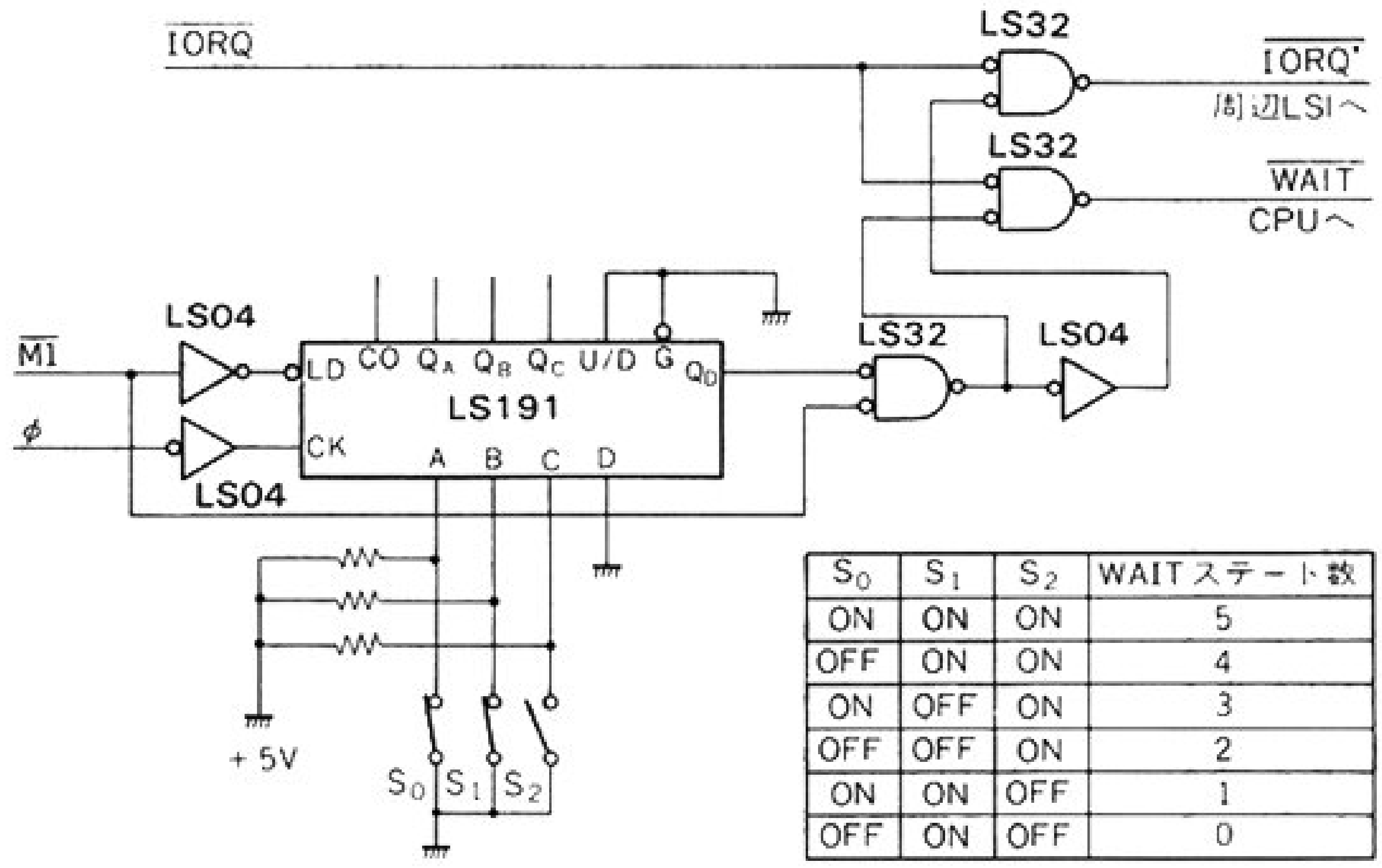
一般的にいって，Z80/Z80A CPUを最高速度で動作させた場合(それぞれ2.5MHz，および4MHz)，周辺LSIを4個までデイジー・チェインすることが可能です．では，5個以上の周辺LSIを使うときにはどのようにしたらよいかを考えてみます．

●CPUのクロック速度を落とす

メモリのスピードやコミュニケーション用のボーレート・ジェネレータ等との関係から，Z80 CPUを目いっぱいの速度で動作させることはごくまれです．例えばZ80A CPUの場合は，18.432MHzの6分の1のクロックである3.072MHzをCPUクロックとして用いることがよくあります．この場合，図3に示した条件は約800nsとなり，5個の周辺LSIをデイジー・チ

エインすることができます.

〈図4〉 $\overline{\text{WAIT}}$によるINTAサイクルの延長



S_0	S_1	S_2	WAITステート数
ON	ON	ON	5
OFF	ON	ON	4
ON	OFF	ON	3
OFF	OFF	ON	2
ON	ON	OFF	1
OFF	ON	OFF	0

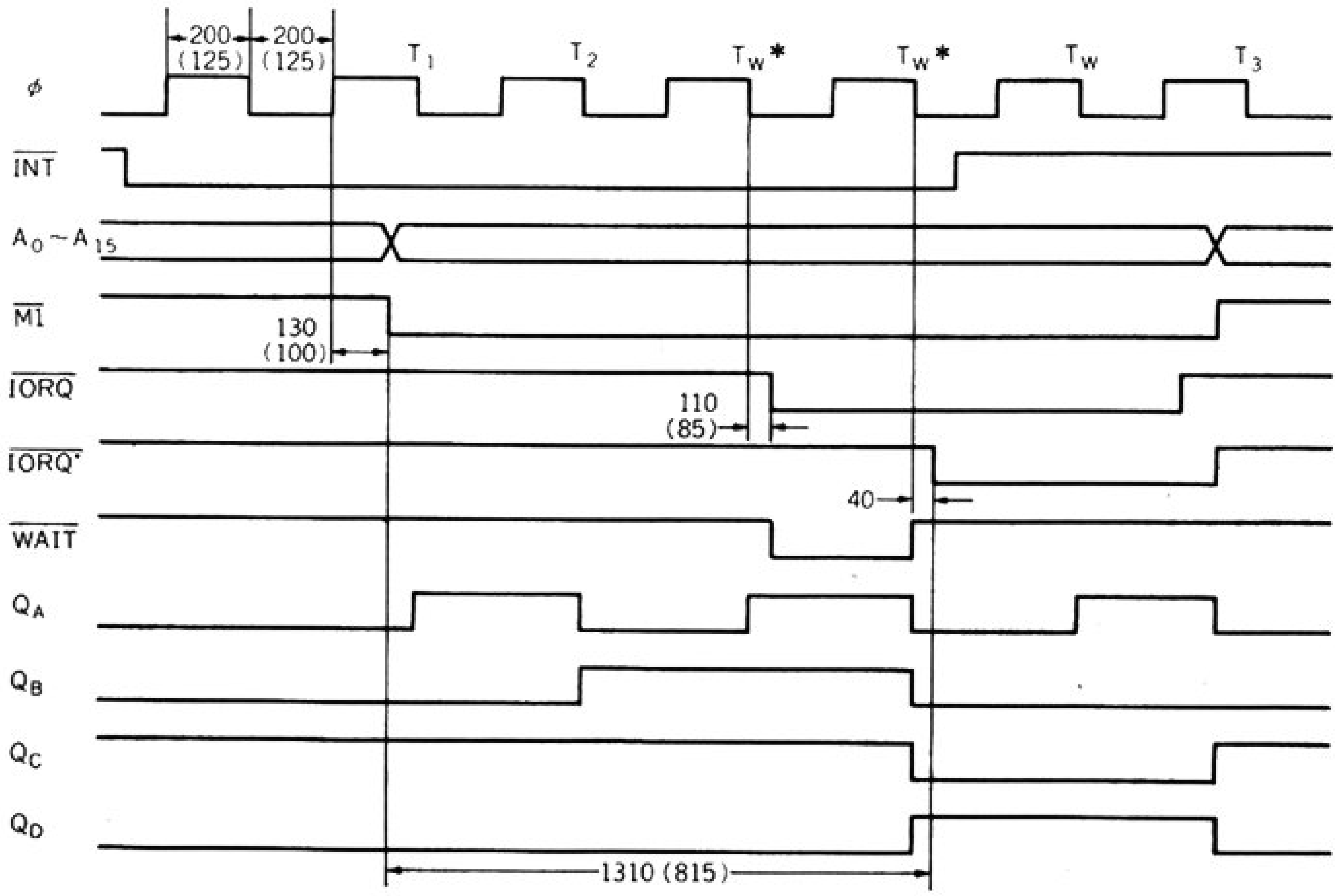
❷ウェイト・ステート(Tw)を追加する

INTAサイクルには, Z80 CPUは自動的に二つのTw*ステートを挿入しますが, 外部からCPUに対して$\overline{\text{WAIT}}$をかけることにより, INTAサイクルを引き延ばすことができます. 図4にこれを行う回路例を示します. また図5に図4のタイミングを示します.

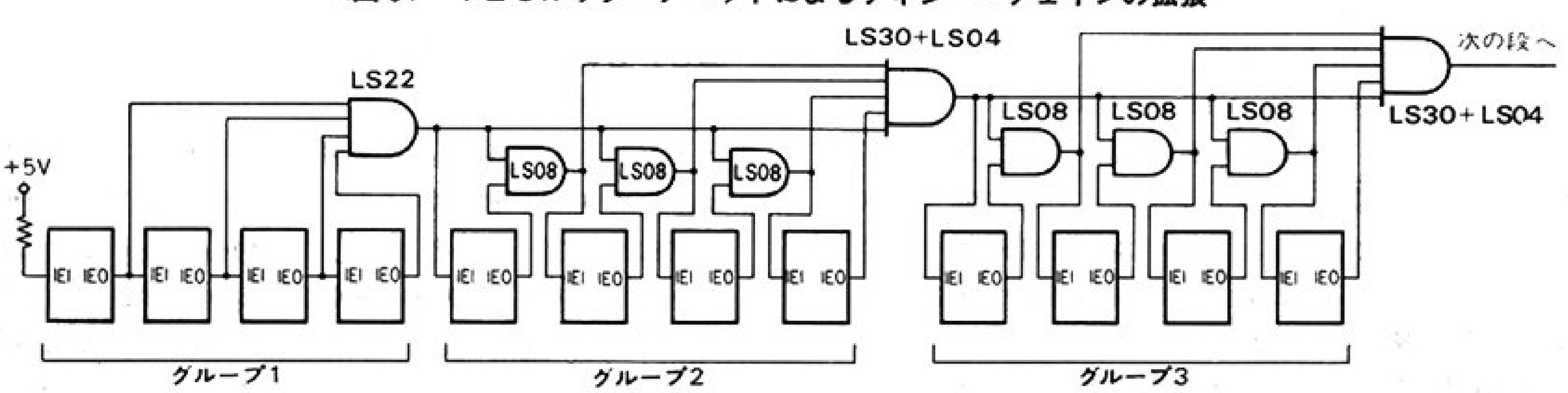
図4の回路ではスイッチS_0~S_2にセットする値により, 0~5個のTwステートを挿入することができます. スイッチS_0~S_2にセットされた値とTwの数との関係も図4中に示されています. この図ではS_0~S_2にセットされた値は4であり, 挿入されるTwの数は1です.

図4の$\overline{\text{IORQ'}}$という信号は, このロジック内部で作られる信号であり, $\overline{\text{IORQ}}$の遅延信号です. 周辺LSIの$\overline{\text{IORQ}}$

〈図5〉
図4のタイミング
(単位;ns)



〈図6〉 IEOルック・アヘッドによるディジー・チェインの拡張



端子にはこの$\overline{IORQ'}$を接続します．

なぜ$\overline{IORQ}$のままではいけないかというと，前述したように周辺LSIは$\overline{M1}$が出ているときの$\overline{IORQ}$により割り込みベクトルをデータ・バス上に乗せますが，IEIの状態が安定していないうちに$\overline{IORQ}$が出されると，いくつかの周辺LSIがデータ・バス・ドライバをイネーブルするような事態(バス競合)が生じる可能性があるからです．

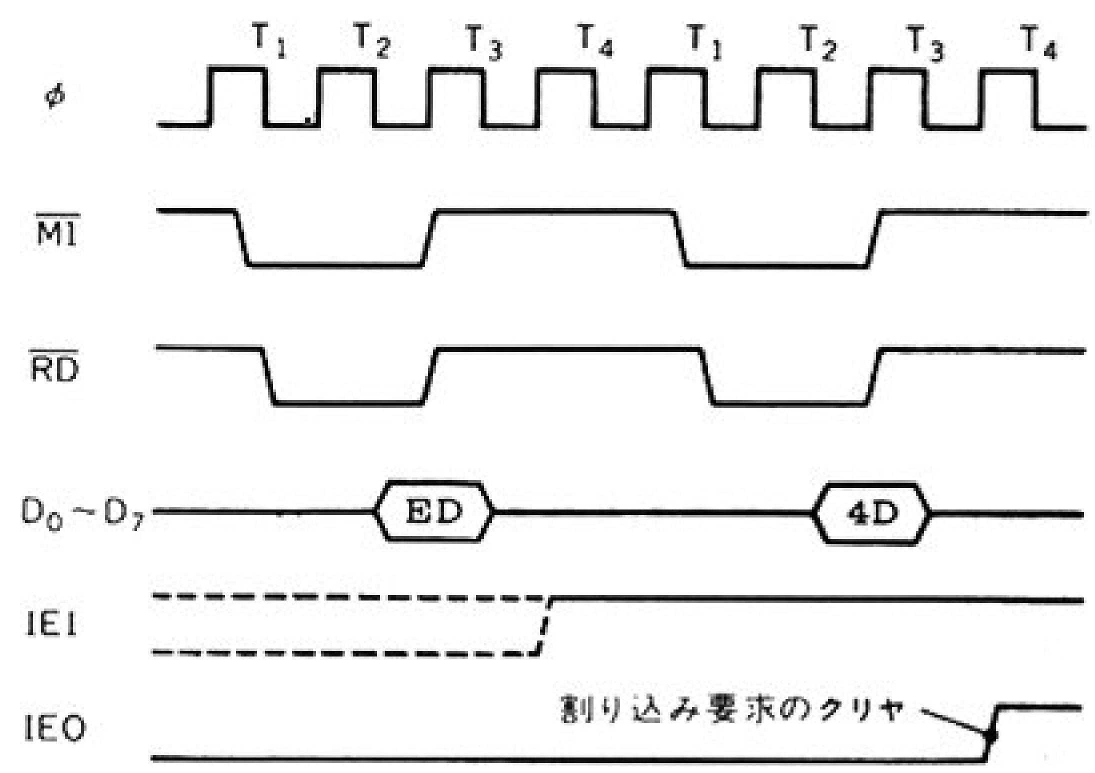
図5からわかるように，Tw ステートを1個挿入することにより，$\overline{M1}$から$\overline{IORQ'}$が出るまでの時間は，$\phi=2.5$MHzの場合で1310ns，$\phi=4$MHz の場合は815nsとなります．図2と比べてみてください．

❸ＩＥＯルック・アヘッドを用いる

アダーにおけるキャリ・ルック・アヘッドと同様の手法を用いて，IEO信号をルック・アヘッドすることにより，IEO信号の伝搬時間を短縮することができます．

図6にIEOルック・アヘッドの例を示します．

〈図7〉 RETIシーケンスのタイミング



割り込み要求のクリヤ

Z80ファミリの周辺ＬＳＩの割り込み要求のクリヤは，CPUがRETI命令を実行することにより行われます．

つまりZ80ファミリの周辺LSIはデータ・バス上のデータを監視しており，あい続く二つの$\overline{M1}$時にED_{16}および$4D_{16}$がデータ・バス上を通過すると，直前に割り込みベクトルを送った周辺LSI(IEIが"H"である)のみが，その割り込み要求をクリヤするようになっています．

このRETIシーケンスのタイミングを図7に示します．このタイミングはZ80ファミリの周辺ＬＳＩ全般にわたって共通です．

補足的なことがらですが，図8にZ80ファミリの周辺ＬＳＩの割り込み制御のロジックを示します．このロジックは，Z80ファミリの周辺ＬＳＩの割り込み制御を理解するうえで大いに役立ちます．データ・バスに接続された256×4ビットのＲＯＭは命令デコーダです．このデコーダはRETI命令(ED_{16}，$4D_{16}$)の他にＯＰコードCB_{16}をデコードしています．この理由は，やじ馬的ではありますが，なかなか趣があります．

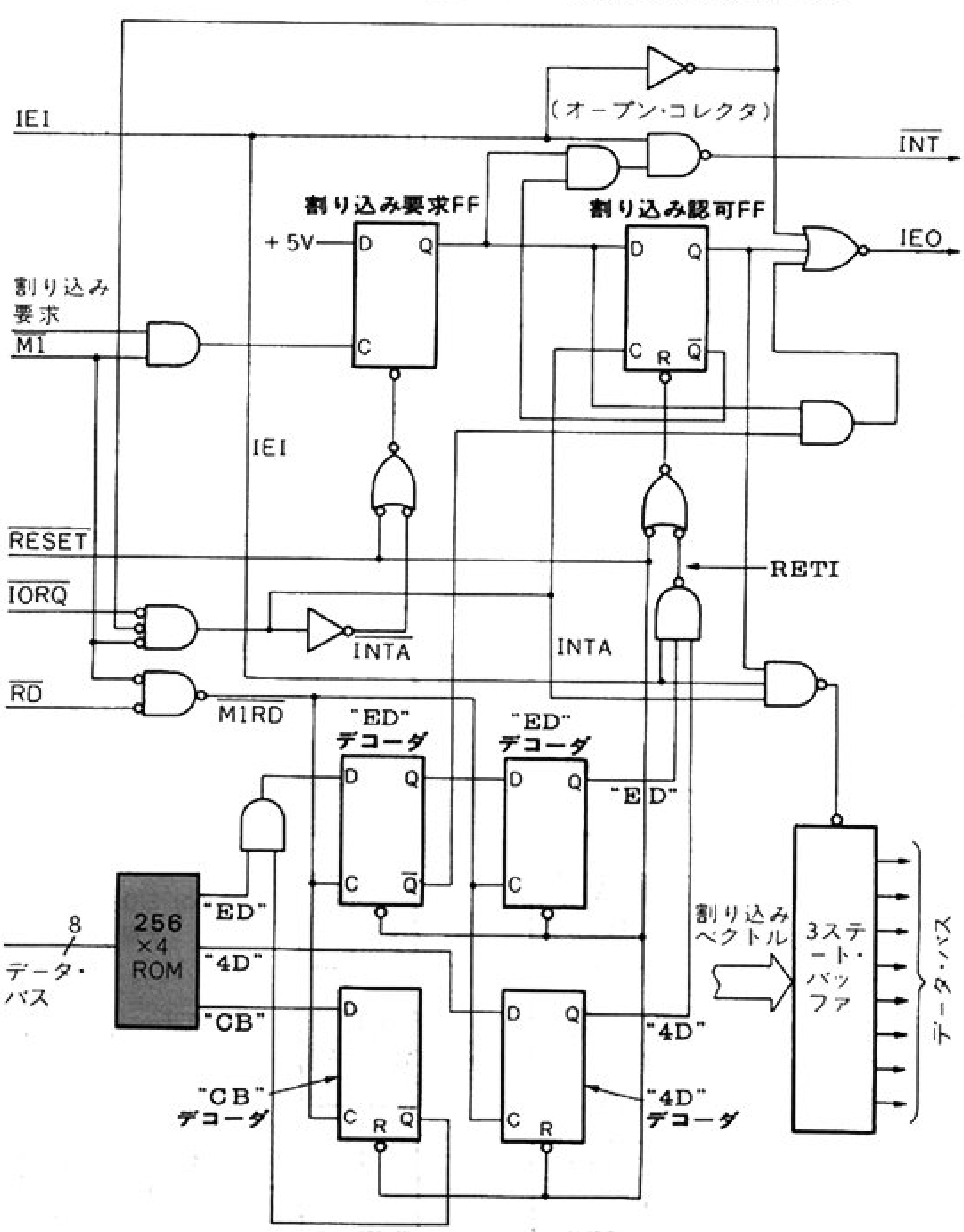
次の二つの命令があい続いて実行されたとしましょう．

SET　5，L　(CB_{16}，ED_{16})

LD　　C，L　($4D_{16}$)

この場合，SET命令の第２ＯＰコード，ED_{16}と次の$4D_{16}$とがあい続く$\overline{M1}$時にデータ・バス上に乗るため，RETI命令

〈図8〉 Z80ファミリ周辺ＬＳＩの割り込み制御ロジック



が実行されたかのような状態となります．このような場合に周辺LSIが誤動作することのないように，OPコードCB_{16}をデコードし，このあとにED_{16}を検出してもRETI命令であるとはみなさないようにするくふうがなされています．

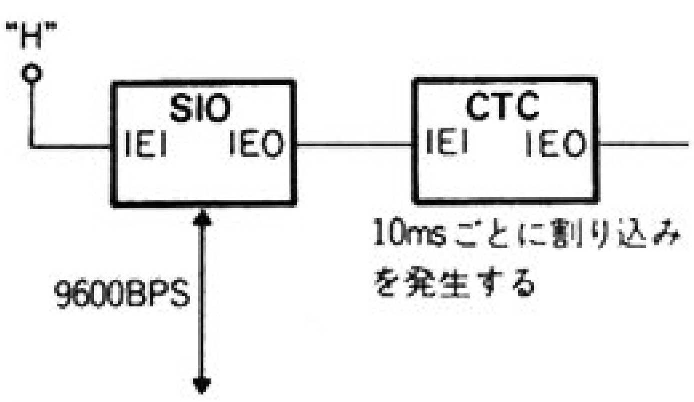
割り込みのネスティング

SIOとCTCが，図9のようなディジー・チェインで接がれている場合を考えてみます．

割り込みの優先度はSIOのほうが上です．SIOは9600 BPSの半2重で動作するものとし，1文字の受信または1文字を送信するごとにCPUに対して割り込みをかけるものとします．

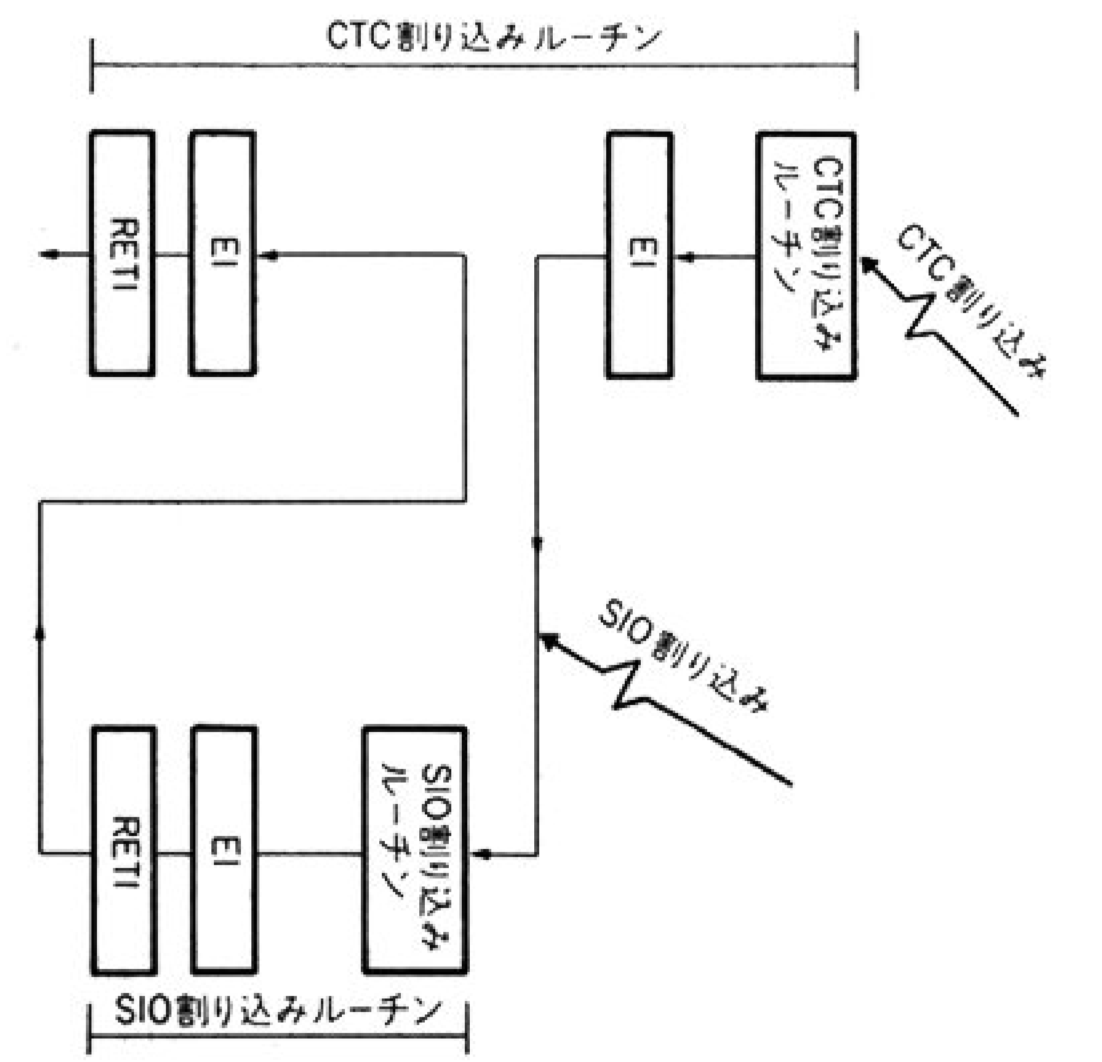
CTCはシステム・クロックとして使われており，10 msごとに割り込みを発生します．CTCの割り込みルーチンでの処理時間はシステムの状況により，1 ms以下である場合もあり，5 msに達することがあるとします．

このような場合，SIOとCTCを同時に動作させると，CTCの割り込みルーチンが動いている場合に，割り込みディスエーブル状態であると，SIOは何回かの割り込みの機会を失ない，結果としてレシーバ・オーバラン，またはトランスミッタ・アンダーランの状態になってしまいます．



〈図9〉割り込みのネスティングを要する例

〈図10〉割り込みのネスティング



このような応用では，CTCの割り込みルーチンは割り込みディスエーブル状態で動くことは許されず，割り込みイネーブル状態で動く必要があります．CTCの割り込みルーチンを割り込みイネーブル状態にすると，同ルーチンが動いている間にもSIOの割り込みが発生し，SIOの割り込みルーチンが終了すると，再びCTCの割り込みルーチンが動きます．

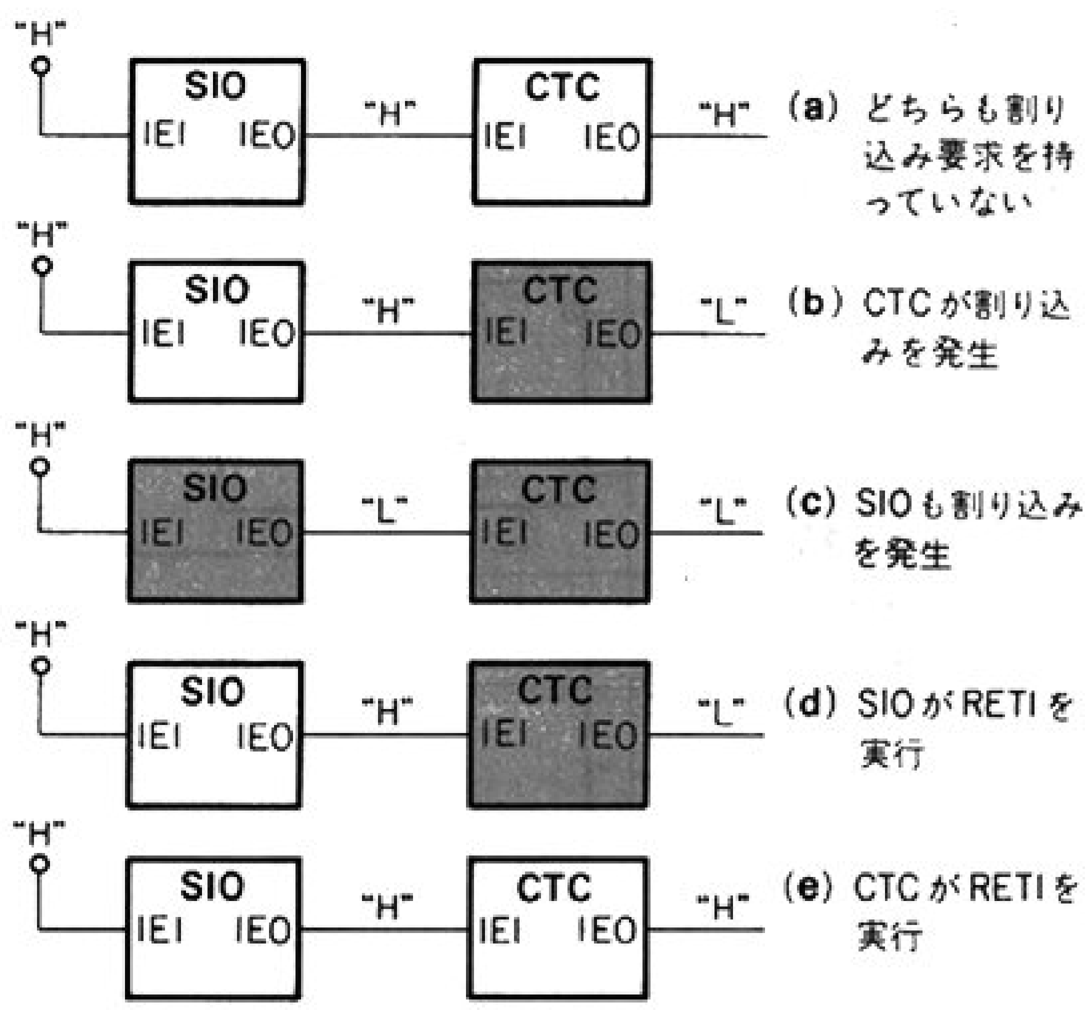
このようにある割り込みを処理している間に，より高位の優先度を持つ割り込みを受け付け，その割り込み処理を終った後に元の割り込み処理を続行することを，割り込みのネスティングといいます．Z 80の割り込みデイジー・チェインは，割り込みのネスティングが容易に行えるような設計になっています．図10に割り込みネスティングの概念を示します．

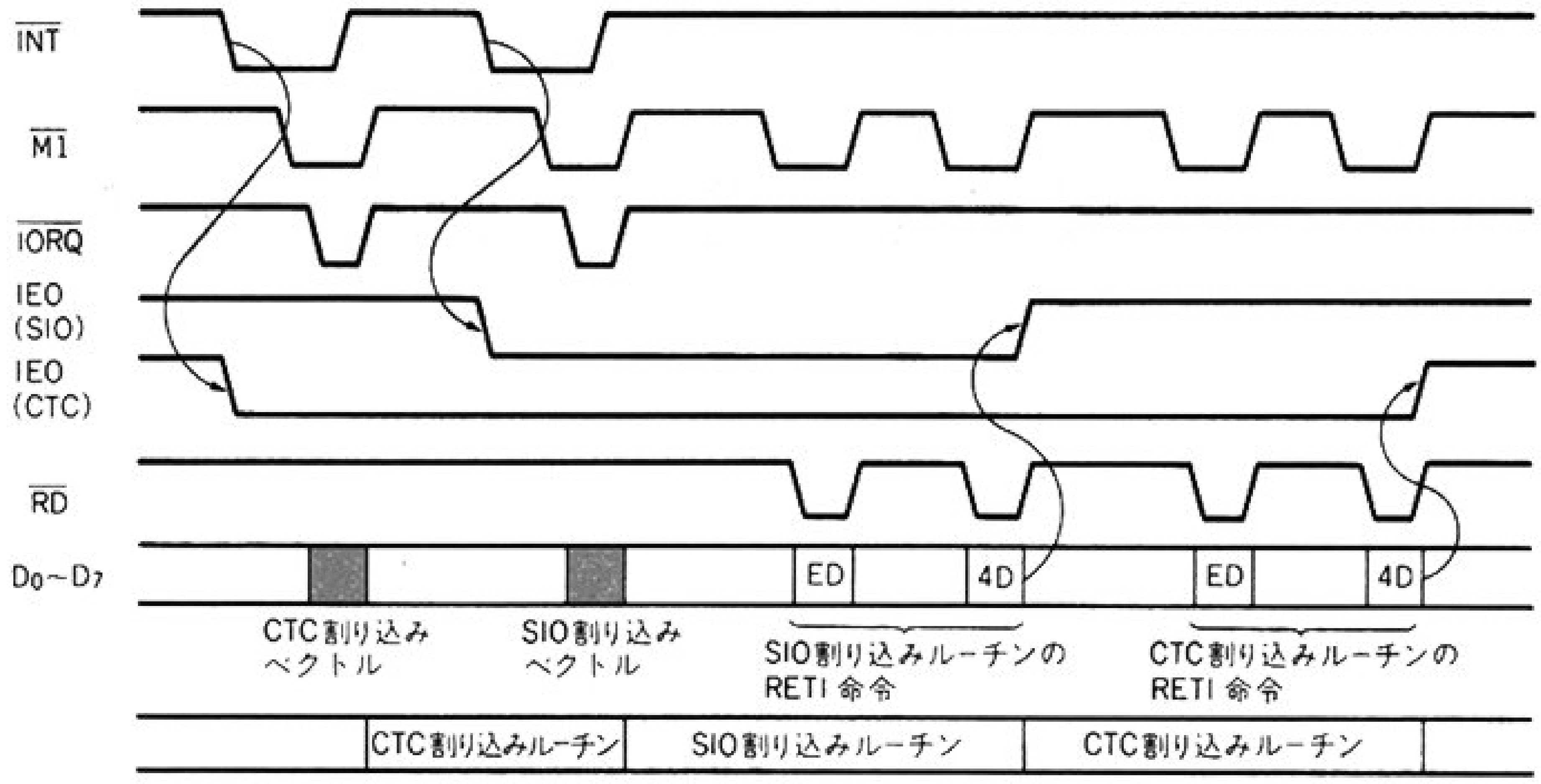
図11により割り込みネスティング時のデイジー・チェインの動きを説明します．まずCTCが割り込みを発生すると，CTCはIEOを"L"にします〔図11(b)〕．CTCの割り込みルーチンが動いているときにSIOが割り込みを発生し，そのIEOを"L"にします〔図(c)〕．SIOの割り込みルーチンがその終りにRETI命令を実行すると，SIOはIEOを"H"にします〔図(d)〕．

ご存知のように，Z 80の周辺LSIはIEIが"H"であるものだけが，RETI命令を認識し，IEOを"H"にするからです．したがってCTCのIEOは相変らず"L"のままです．最後にCTCの割り込みルーチンがRETI命令を実行すると，CTCはそのIEOを"H"にし，元の状態に戻ります．以上の過程をタイミング・チャートにしたものを図12に示します．

割り込みルーチンでは，通常レジスタのセーブ(プッシュ)，およびリストア(ポップ)を行いますが，割り込みのネスティングを行ったからといって，とくにそれ

〈図11〉割り込みネスティング時のデイジー・チェイン





〈図 12〉
割り込みネスティングのタイミング

に注意を払う必要はありません．サブルーチンのネスティングの場合と同じようにプッシュ・ダウン・スタックがうまく操作されるからです．

Appendix 3 Z80と64KダイナミックRAMとのインターフェース

64 KダイナミックRAM(以下DRAM)の価格低下はすばらしく，秋葉原あたりにおける値段をみると，16 K DRAMが1個数100円程度で，64 K DRAMは1個1000円くらいです．つまりビット当たりのコストは，64 K DRAMのほうが低くなっています．

また最近ではZ 80 CPUに64 Kバイト以上のメモリをぶらさげるようなシステムも，まれではありません．仮りに64 Kバイトのメモリを作るとすると，16 K DRAMでは32個のチップが必要ですが，64 K DRAMを使えばわずか8個のチップですみます．

これはプリント基板のボード面積の縮小につながり，大きなコスト・メリットになります．さらに必要な電源の種類をみてみると，16 K DRAMは3種類(+12 V, +5 V, −5 V)であるのに対して，64 K DRAMのそれは単一5 V電源ですみます．これも設計者にとって大きなメリットです．したがってZ 80を使う者にとって64 K DRAMを用いる技術は，今では必須のことがらになっています．

64K DRAMの動作概要

図1に64 K DRAMのピン配置図を示します．64 K DRAMは16ピンDIPであり，8本のアドレス入力A_0～A_7，$\overline{RAS}$(Row Address Strobe)，$\overline{CAS}$(Column Address Strobe)，$\overline{WE}$(Write Enable)，D_{IN}(Data Input)，D_{OUT}(Data Output)，*Vcc*(電源：+5 V)，ならびに*Vss*(グラウンド)の合計15ピンが使われており，ピン1はNC(Not Connected)で，使用されていません．

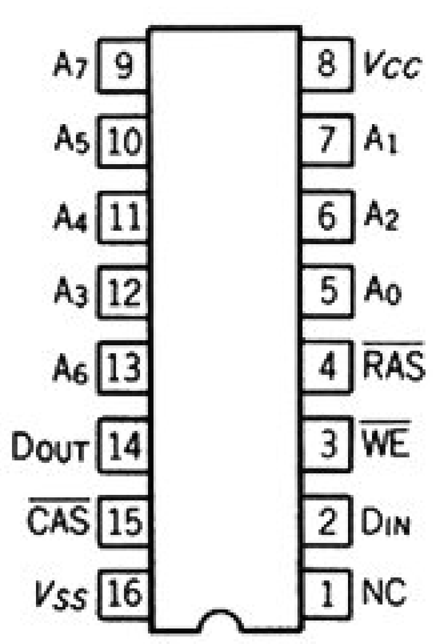
64 K DRAMを機能的に見ると，図2のようになります．この図を用いて64 K DRAMの動作を説明します．

64 K DRAMは64 K(65536)×1ビット構成になっています．したがって，本来ならば16本のアドレス・ラインを必要としますが，64 K DRAMでは8本しかアドレス・ラインがありませんので，アドレスを行(ぎょう)アドレス(Row Address)と列(れつ)アドレス(Column Address)の8本ずつに分けて入力します．

64 K DRAMからデータを読む場合には，まず行アドレスをA_0～A_7上に乗せ，$\overline{RAS}$を“L”にします．これによりDRAMは行アドレスを内部にラッチします．次に列アドレスをA_0～A_7上に乗せ，$\overline{CAS}$を“L”にしますと，DRAMは列アドレスを内部にラッチします．その後DRAMは内部でメモリ・セルのアクセスを開始し，メモリ・データをDout上に出力します．

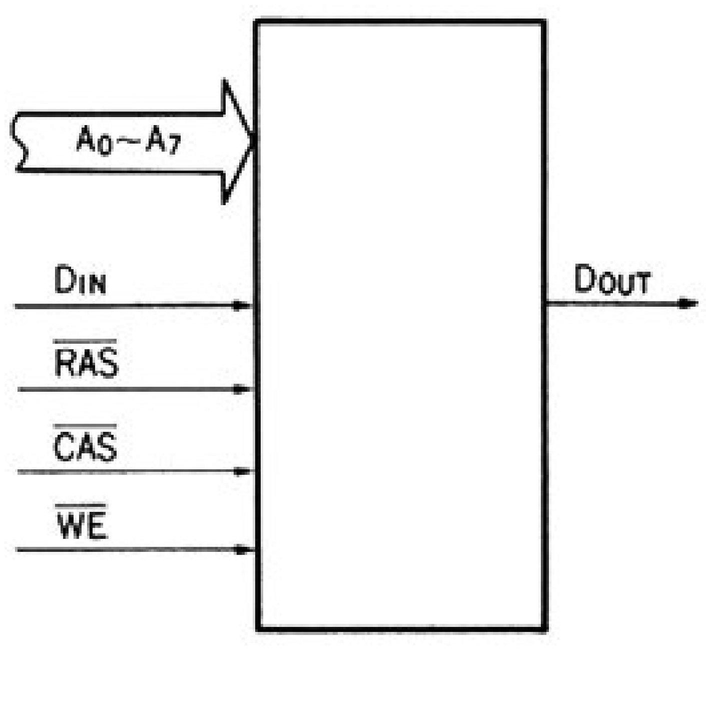
DRAMに対してデータを書き込むときには，$\overline{CAS}$を“L”にする前に，D_{IN}上に入力データを乗せ，$\overline{WE}$を“L”にすると，行アドレスおよび列アドレスにより選択されたメモリ・セルにD_{IN}上のデータが書き込まれます．

〈図1〉64 K DRAMのピン配置



A_0～A_7	Address	アドレス入力
$\overline{CAS}$	Column Addres Strobe	カラム・アドレス・ストローブ
D_{IN}	Data Input	データ入力
D_{OUT}	Data Output	データ出力
$\overline{RAS}$	Row Address Strobe	ロー・アドレス・ストローブ
$\overline{WE}$	Write Enable	ライト・イネーブル
Vcc	Power Supply	電源（+5V)
Vss	Ground	グラウンド(0V)

〈図2〉64 K DRAMの信号線を機能的に見た図



項目		記号	min	typ	max	単位
電源電圧		V_{CC}	4.5	5.0	5.5	V
		V_{SS}	0	0	0	V
入力電圧		V_{IH}	2.4	—	6.5	V
		V_{IL}	−1.0	—	0.8	V

〈図3〉
64 K DRAMの電気的特性
(日立HM 4864 AP)

■DC電気的特性(V_{CC}=5V±10%, V_{SS}=0V, T_a=0～+70℃)

項目	記号	測定条件	HM4864A/P-12		HM4864A/P-15		HM4864A/P-20		単位	注
			min	max	min	max	min	max		
動作電流	I_{CC1}	$\overline{RAS}$, $\overline{CAS}$サイクル, t_{RC}=min	—	55	—	50	—	44	mA	1
スタンバイ電流	I_{CC2}	$\overline{RAS}$=V_{IN}, D_{OUT}=ハイ・インピーダンス	—	3.5	—	3.5	—	3.5	mA	
リフレッシュ電流	I_{CC3}	$\overline{RAS}$サイクル, $\overline{CAS}$=V_{IH}, t_{RC}=min	—	42	—	38	—	33	mA	
スタンバイ電流	I_{CC5}	$\overline{RAS}$=V_{IH}, D_{OUT}=イネーブル	—	5.5	—	5.5	—	5.5	mA	1
ページ・モード電流	I_{CC6}	$\overline{RAS}$=V_{IL}, $\overline{CAS}$サイクル, t_{PC}=min	—	38	—	35	—	31	mA	1
入力漏洩電流	I_{LI}	V_{in}=0～6.5V, 他全入力=0V	−10	10	−10	10	−10	10	μA	
出力漏洩電流	I_{LO}	V_{out}=0～5.5V, D_{OUT}=ディスエーブル	−10	10	−10	10	−10	10	μA	2
出力電圧	V_{OH}	I_{out}=−5mA	2.4	V_{CC}	2.4	V_{CC}	2.4	V_{CC}	V	
	V_{OL}	I_{out}=4.2mA	0	0.4	0	0.4	0	0.4	V	

注)1. I_{CC}はデバイスが選択されたときの出力負荷条件で決まります. I_{CC}のmax値は出力開放状態です. 2. I_{LO}漏洩電流のみの値です.

■容　量(V_{CC}=5V±10%, T_a=25℃)

項目		記号	測定条件	min	max	単位	注
入力容量	A_0～A_7, D_{IN}	C_{IN}		—	5	pF	1
	$\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$			—	10		
出力容量	D_{OUT}	C_{OUT}		—	7	pF	1,2

注)1. ブントン・メータ, あるいはそれに等価な測定方法によって測定した値. 2. D_{OUT}をディスエーブルするために$\overline{CAS}$=V_{IH}にして測定した値.

行アドレスおよび列アドレスという呼び名は, DRAMの内部構成が行列構造になっていることからきていますが, DRAMのユーザーである私たちにとっては, そんな呼び名などにあまり関心を持つ必要はなく, 要は両アドレスにより, 16ビットのアドレスをDRAMにたたき込めばよいわけです.

したがって, 行アドレスとしてCPUのA_{15}～A_8を用い, 列アドレスとしてA_7～A_0を用いてもよいし, その逆であってもかまいません. 極端なことをいうと, 行アドレスとしてA_0, A_2, A_4, …, A_{14}, 列アドレスとしてA_1, A_3, A_7, …, A_{15}というように奇・偶に分けてもかまいません.

しかしながら, DRAMではリフレッシュ動作は行アドレスにより行われ, リフレッシュ・アドレスとしてA_0～A_6を用いるのがふつうなので, A_0～A_7を行アドレス, A_8～A_{15}を列アドレスとすることがよく行われています. 実際Z 80 CPUもリフレッシュ・アドレスをA_0～A_6上に出力してきます.

64 K DRAMは+5 V単一電源で動作し, その入出力信号はTTLコンパチブルです. またD_{OUT}の"L"レベルにおける最大値は, 負荷電流が4.2 mAであるときに, 0.4 Vですので, SタイプのTTLを2個ドライブすることができます. 図3に64 K DRAMの電気的特性を示します.

64K DRAMのタイミング

64 K DRAMは動作サイクルとして以下の六つのサイクルを持っています.

・リード・サイクル
・ライト・サイクル
・リード/モディファイ/ライト・サイクル
・ページ・モード・リード・サイクル
・ページ・モード・ライト・サイクル
・リフレッシュ・サイクル

リード/モディファイ/ライト・サイクルというのは, 1メモリ・サイクル中でリード動作とライト動作を同時にやってしまうサイクルのことです. ページ・モード・リード・サイクルは, 1回の$\overline{RAS}$に対して複数個の$\overline{CAS}$を出すことにより, 同一行アドレス中のデータを高速に読み出すためのサイクルです.

またページ・モード・ライト・サイクルは1回の$\overline{RAS}$に対して複数個の$\overline{CAS}$を出すことにより, 同一行アドレス中にデータを高速に書き込むためのサイクルです.

これらのサイクルもDRAMを種々の用途に適用する場合には絶対に知っておかなければならないものですが, 本稿では64 K DRAMとZ 80 CPUとのインター

フェースに主眼をおいていますので，説明を省略します．

■リード・サイクル

図4に64 K DRAMのリード・サイクルのタイミングを示します．リード・サイクルは行アドレスをA_0～A_7上に乗せ，$\overline{\text{RAS}}$を"L"にすることから始まります．行アドレスは$\overline{\text{RAS}}$が立ち下がる時点よりもt_{ASR}だけ前に確定しておく必要があります(セットアップ・タイム)．

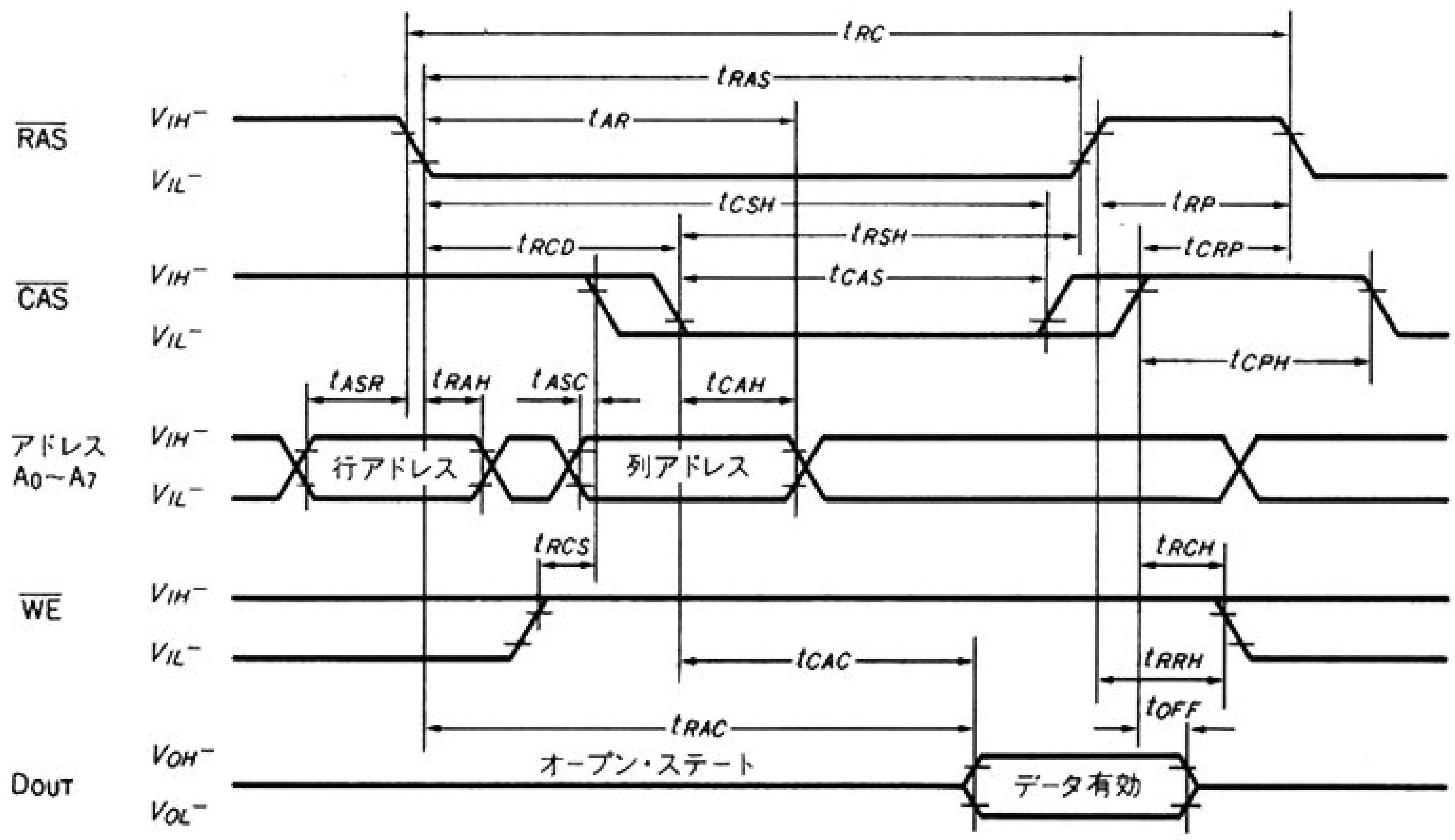
また行アドレスは$\overline{\text{RAS}}$が立ち下がってからt_{RAH}だけの期間保持しておく必要があります(ホールド・タイム)．次にA_0～A_7上に列アドレスを乗せ，$\overline{\text{CAS}}$を"L"にします．列アドレスのセットアップ・タイムはt_{ASC}であり，ホールド・タイムはt_{CAH}です．

$\overline{\text{RAS}}$を"L"にしてから$\overline{\text{CAS}}$を"L"にするまでの時間はt_{RCD}であり，これには最小値と最大値の規定があります．最小値は絶対に守らなければなりませんが，最大値のほうは守る必要はありません．t_{RCD}を最大値以上にした場合は，アクセス・タイムがそれだけ遅くなるだけです．

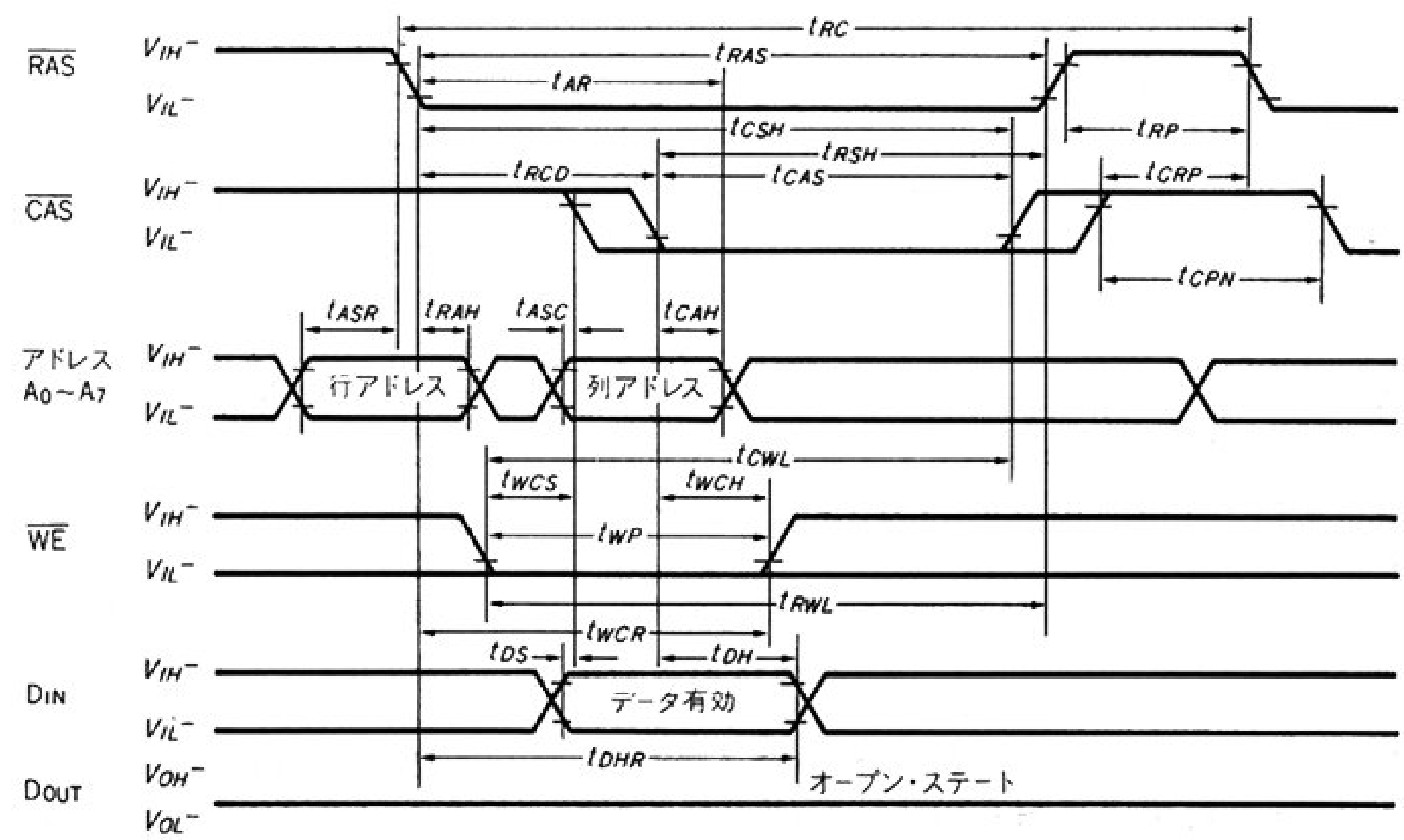
出力データはD_{OUT}上に出力されます．$\overline{\text{RAS}}$が立ち下がってからD_{OUT}上に出力データが出てくるまでの時間をアクセス・タイムといいますが，その時間はt_{RCD}が最大値以内にあるか否かにより異なります．

t_{RCD}が最大値以内であれば，アクセス・タイムはt_{RAC}により規定されています．またt_{RCD}が最大値を越えているときには，アクセス・タイムはt_{RCD}＋t_{CAC}となり

〈図4〉
リード・サイクルのタイミング
(HM 4864 AP)



〈図6〉
ライト・サイクルのタイミング
(HM 4864 AP)



ます．

$\overline{\text{RAS}}$および$\overline{\text{CAS}}$のパルス幅の最小値は t_{RAS} および t_{CAS} として決められています．t_{RAS} には最大値の制限が設けられていますが，ページ・モードを使わない限り，とくに注意を払う必要はありません．

1回リード・サイクルを実行し，次のサイクルを開始するには，t_{RC}だけの時間を必要とします．この時間のことをランダム・リード/ライト・サイクル・タイムと呼びます．また$\overline{\text{RAS}}$にはRASプリチャージ・タイムという規定があり，$\overline{\text{RAS}}$が立ち上がってから再度$\overline{\text{RAS}}$が立ち下がるまでの最小時間が決められています．

図5に64 K DRAMのタイミング定数を示します．この図はHM-6864 AP(日立)の場合ですが，NECや三菱の64 K DRAMもほとんど同じ特性を持っています．三者の間に相異点があったとしても，セットアップ・タイムやホールド・タイムに5 ns程度の差があるくら

〈図5〉64 K DRAMのタイミング定数(HM 4864 AP)

■**AC特性**($V_{CC}=5V\pm10\%$, $V_{SS}=0V$, $T_a=0\sim+70^\circ C$) 1), 2)

項目	記号	HM4864A/P-12		HM4864A/P-15		HM4864A/P-20		単位	注
		min	max	min	max	min	max		
ランダム・リード／ライト・サイクル時間	t_{RC}	220	—	260	—	330	—	ns	
リード／ライト・サイクル時間	t_{RWC}	245	—	280	—	345	—	ns	
ページ・モード・サイクル時間	t_{PC}	120	—	145	—	190	—	ns	
RASからのアクセス時間	t_{RAC}	—	120	—	150	—	200	ns	4,6
CASからのアクセス時間	t_{CAC}	—	60	—	75	—	100	ns	5,6
出力バッファ・ターンオフ遅延	t_{OFF}	—	35	—	40	—	50	ns	7
トランジション時間(上昇/下降)	t_r	3	35	3	35	3	50	ns	3
RASプリチャージ時間	t_{RP}	90	—	100	—	120	—	ns	
RASパルス幅	t_{RAS}	120	10000	150	10000	200	10000	ns	
RASホールド時間	t_{RSH}	60	—	75	—	100	—	ns	
CASパルス幅	t_{CAS}	60	10000	75	10000	100	10000	ns	
CASホールド時間	t_{CSH}	120	—	150	—	200	—	ns	
RAS・CAS遅延時間	t_{RCD}	25	60	25	75	30	100	ns	8
CAS・RASプリチャージ時間	t_{CRP}	−10	—	−10	—	−10	—	ns	
ロウ・アドレス・セットアップ時間	t_{ASR}	0	—	0	—	0	—	ns	
ロウ・アドレス・ホールド時間	t_{RAH}	15	—	15	—	20	—	ns	
カラム・アドレス・セットアップ時間	t_{ASC}	0	—	0	—	0	—	ns	
カラム・アドレス・ホールド時間	t_{CAH}	20	—	25	—	30	—	ns	
カラム・アドレス・ホールド時間(RASに対して)	t_{AR}	80	—	100	—	130	—	ns	
リード・コマンド・セットアップ時間	t_{RCS}	0	—	0	—	0	—	ns	
リード・コマンド・ホールド時間	t_{RCH}	0	—	0	—	0	—	ns	
ライト・コマンド・ホールド時間	t_{WCH}	40	—	45	—	55	—	ns	
ライト・コマンド・ホールド時間(RASに対して)	t_{WCR}	100	—	120	—	155	—	ns	
ライト・コマンド・パルス幅	t_{WP}	40	—	45	—	55	—	ns	
ライト・コマンド・RASリード時間	t_{RWL}	40	—	45	—	55	—	ns	
ライト・コマンド・CASリード時間	t_{CWL}	40	—	45	—	55	—	ns	
データ入力セットアップ時間	t_{DS}	0	—	0	—	0	—	ns	9
データ入力ホールド時間	t_{DH}	40	—	45	—	55	—	ns	9
データ入力ホールド時間(RASに対して)	t_{DHR}	100	—	120	—	155	—	ns	
CASプリチャージ時間(ページ・モード・サイクル)	t_{CP}	50	—	60	—	80	—	ns	
リフレッシュ周期	t_{REF}	—	2	—	2	—	2	ms	
WEコマンド・セットアップ時間	t_{WCS}	0	—	0	—	0	—	ns	10
CAS・WE遅延	t_{CWD}	40	—	45	—	55	—	ns	10
RAS・WE遅延	t_{RWD}	100	—	120	—	155	—	ns	
RASプリチャージ・CASホールド時間	t_{RPC}	0	—	0	—	0	—	ns	
リード・モディファイ・ライト・ホールド時間	t_{RRH}	10	—	10	—	10	—	ns	
CASプリチャージ時間	t_{CPN}	30	—	35	—	45	—	ns	

注) 1．AC測定はt_r=5nsとする．

2．電源投入後あるいは$\overline{\text{RAS}}$の期間が2ms以上引き延ばされる場合には，必ず8サイクルを実行した上で正規のメモリ動作に入るようシステム設計してください．

3．V_{IH}(min)とV_{IL}(max)は入力信号の測定タイミングの参照レベルです．トランジション時間はV_{IH}からV_{IL}への立ち下がり時間，またはその逆の立ち上がり時間です．

4．$t_{RCD}=t_{RCD}$(max)とします．t_{RCD}がこの表の最大値より大きい場合，t_{RAC}は規定値を越えます．

5．$t_{RCD}=t_{RCD}$(max)とします．

6．2TTL+100pFに等価な負荷回路で測定．

7．t_{OFF}(max)は出力が開放状態に達し，出力電圧レベルを参照できなくなった場合の時間で定義します．

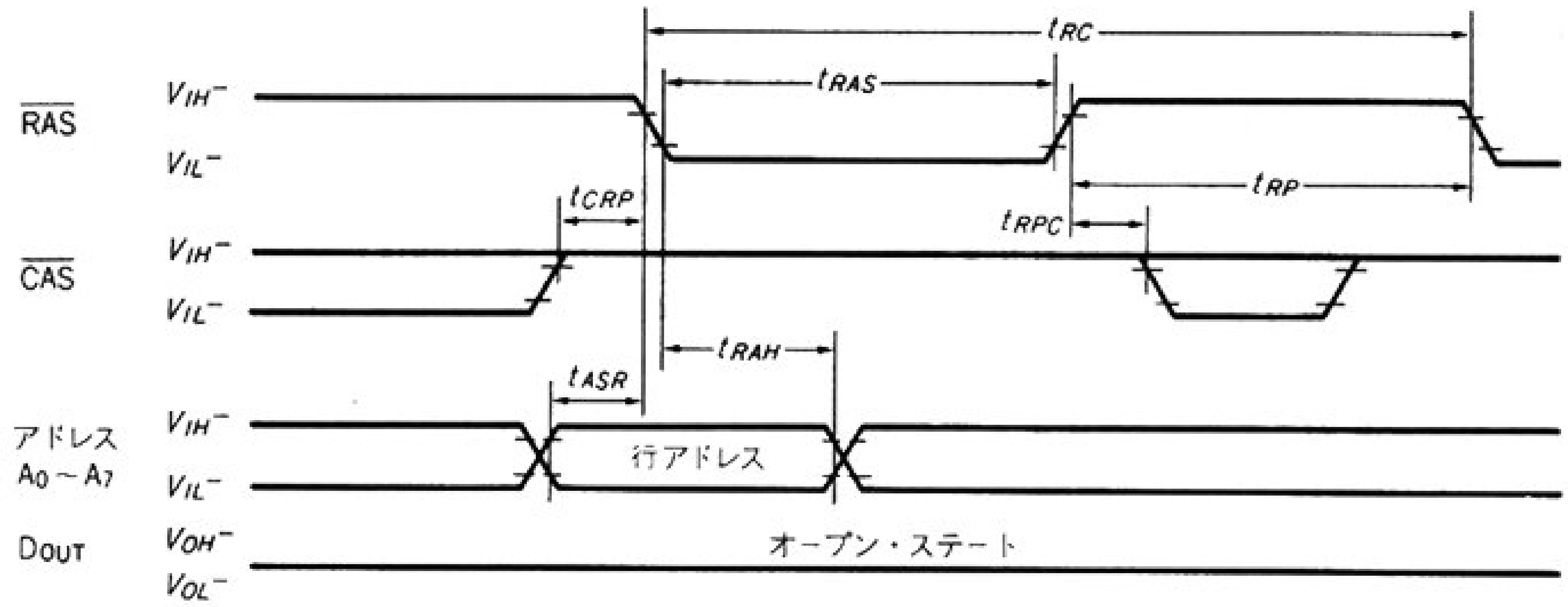
8．t_{RCD}がt_{RCD}(max)より大きい場合，アクセス時間はt_{CAC}によって規定されます．

9．$\overline{\text{CAS}}$リード・エッジと$\overline{\text{WE}}$リード・エッジのいずれか遅いほうのリード・エッジから定義されます．

10．$t_{WCS}\geq t_{WCS}$(min)の場合，このサイクルはアーリ・ライト・サイクルとなります．データ出力端子はこのサイクルの間，ハイ・インピーダンス状態が保たれます．

$t_{CWD}\geq t_{CWD}$(min)かつ$t_{RWD}\geq t_{RWD}$(min)の場合，選択されたアドレスのデータがデータ出力端子に出力され，入力データが選択されアドレスへ書き込まれます(リード・モディファイ・ライト・サイクルに適用される)．

〈図7〉
リフレッシュ・サイクルのタイミング



いのことであり，よほどクリティカルな設計をしない限り，差異はないといってもよいでしょう．

■ライト・サイクル

図6に64 K DRAMのライト・サイクルのタイミングを示します．このサイクルにおける$\overline{\text{RAS}}$, $\overline{\text{CAS}}$ならびにアドレスの関係はリード・サイクルの場合と同じです．ライト・サイクルでは$\overline{\text{CAS}}$を“L”にする前に，D_{IN}上に書き込みデータを乗せ，$\overline{\text{WE}}$を“L”にします．D_{IN}上のデータは$\overline{\text{CAS}}$の立ち下がりより，t_{DS}だけ前に確定し，$\overline{\text{CAS}}$の立ち下がり後t_{DH}の期間データを保持しておく必要があります．

また，$\overline{\text{WE}}$は$\overline{\text{CAS}}$の立ち下がりより，t_{WCS}だけ前に“L”になっていなければならず，$\overline{\text{CAS}}$の立ち下がり後t_{WCH}の期間その状態を保持しておかなければなりません．このほか$\overline{\text{WE}}$にはパルス幅の規定があり，最小値がt_{WP}で決められています．

■リフレッシュ・サイクル

DRAMはメモリ・セル中のキャパシタに電荷を蓄えることにより，データを記憶しますが，このキャパシタに蓄えられた電荷は時間とともに放電してしまい，何もしないでいると記憶していたデータは失なわれてしまいます．このためDRAMでは一定時間ごとにメモリ・セルのデータを書き直すことにより，記憶データを保持する過程を必要とします．この過程のことをDRAMのリフレッシュと呼びます．

64 K DRAMでは各メモリ・セルを2 msに1回の割で，リフレッシュする必要があります．ただし，64 K DRAMでは2 msごとに128行(Row)に対してリフレッシュ動作を行うことにより，64 Kビットすべてのメモリ・セルがリフレッシュされるようになっています．

図7に64 K DRAMのリフレッシュ・サイクルのタイミングを示します．このサイクルは単純で，リフレッシュすべき行のアドレスを$A_0 \sim A_6$上に乗せ，$\overline{\text{RAS}}$を“L”にするだけのことです．この場合，A_7の状態は“Don't Care”です．

またDRAMはメモリ・リード・サイクル，またはメモリ・ライト・サイクルが実行されると，これらのサイクルにおいて指定された行アドレスのセルに対してリフレッシュ動作を自動的に行います．

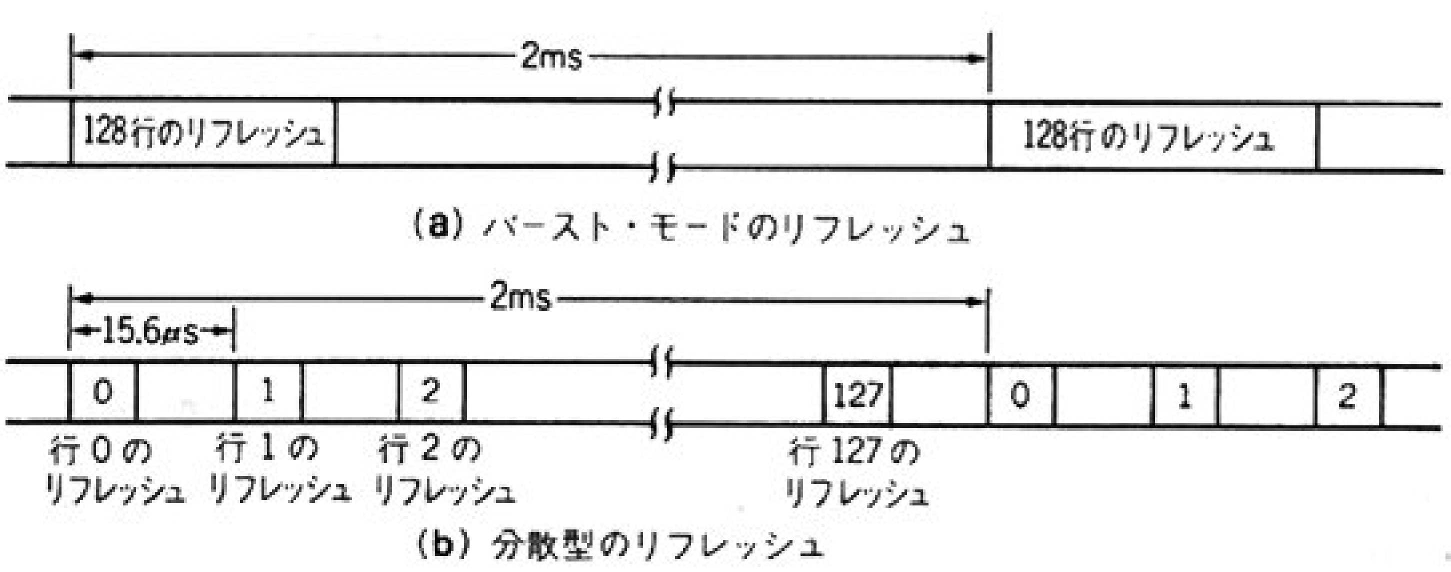
したがって，CRTディスプレイのリフレッシュ・メモリなどのように，周期的に128行のメモリに対してメモリ・リード・サイクルが実行されるような場合には，リフレッシュ・サイクルを意識しないでも，自動的にリフレッシュ動作が行われます．

DRAMのリフレッシュを行うには，タイミング的に2通りの方法，

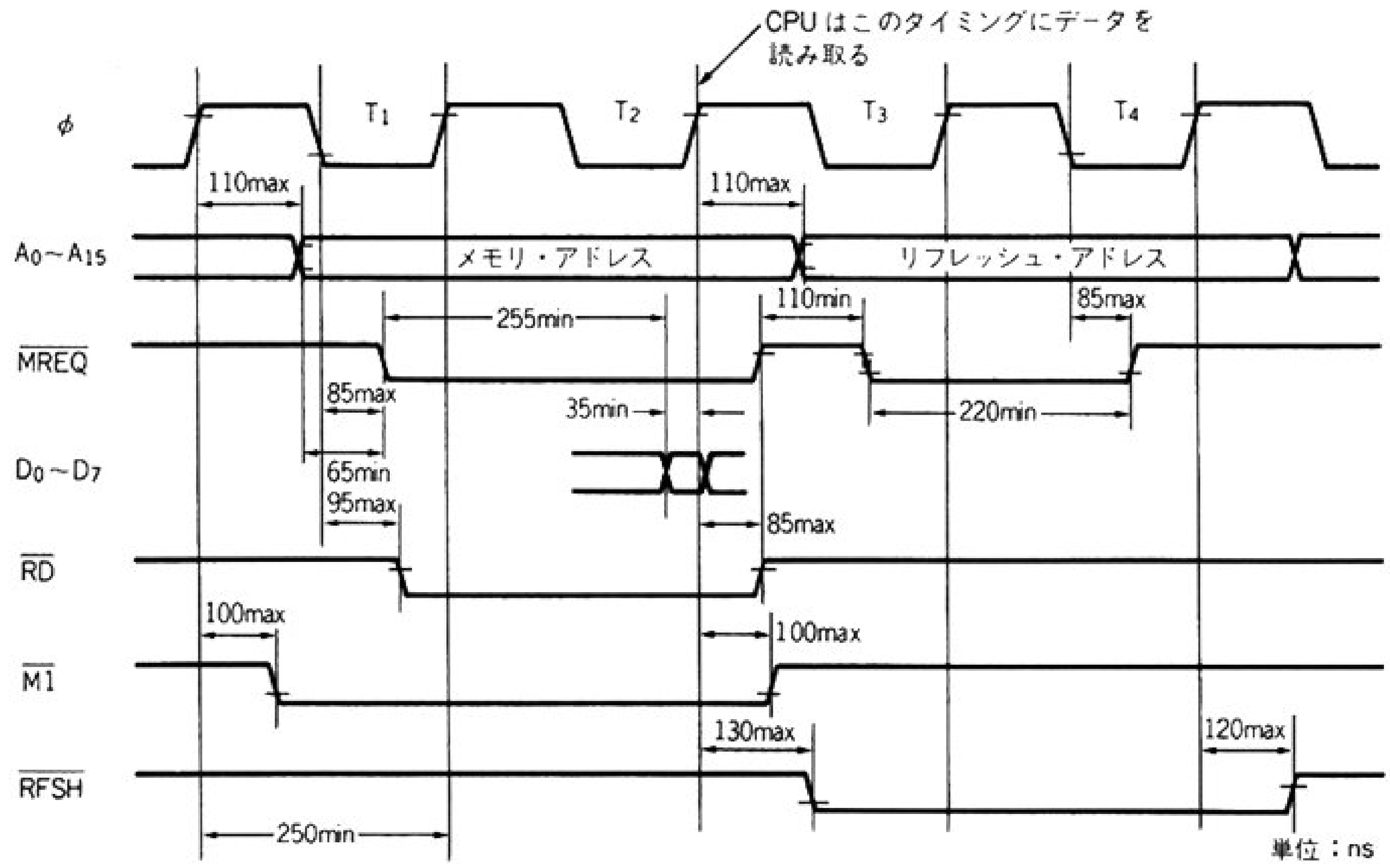
① バースト・モード
② 分散型

があります．図8にこれらのリフレッシュ法を示します．

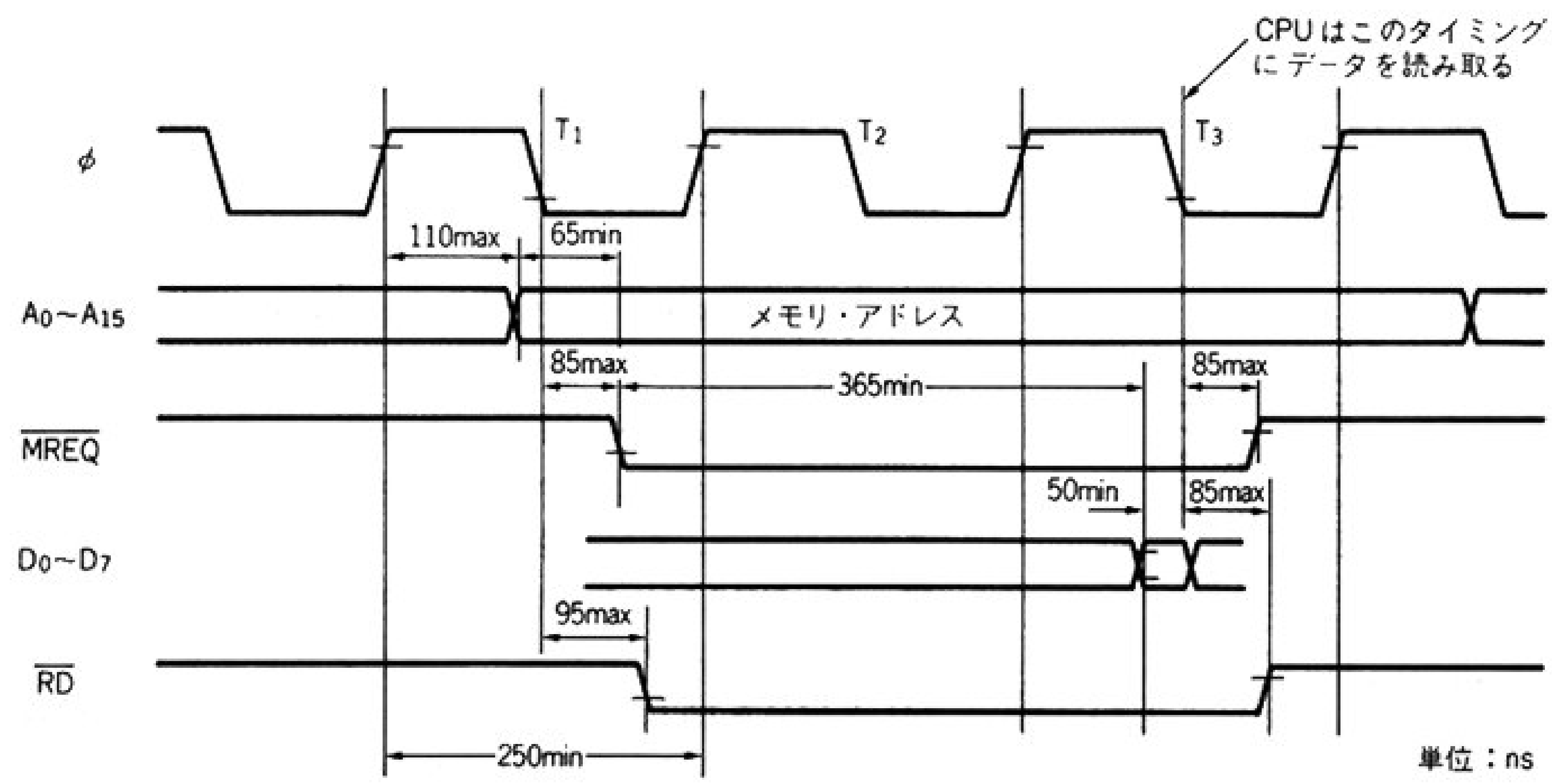
バースト・モードのリフレッシュは，2 msに1回の割でCPUを待たせ(WAIT)，その間に128行全体に対してリフレッシュ動作を行います．分散型のリフレッシュでは2 msの期間を128に分け，約15.6 μs(2000 μs÷128)に1回の割合で各行のリフ



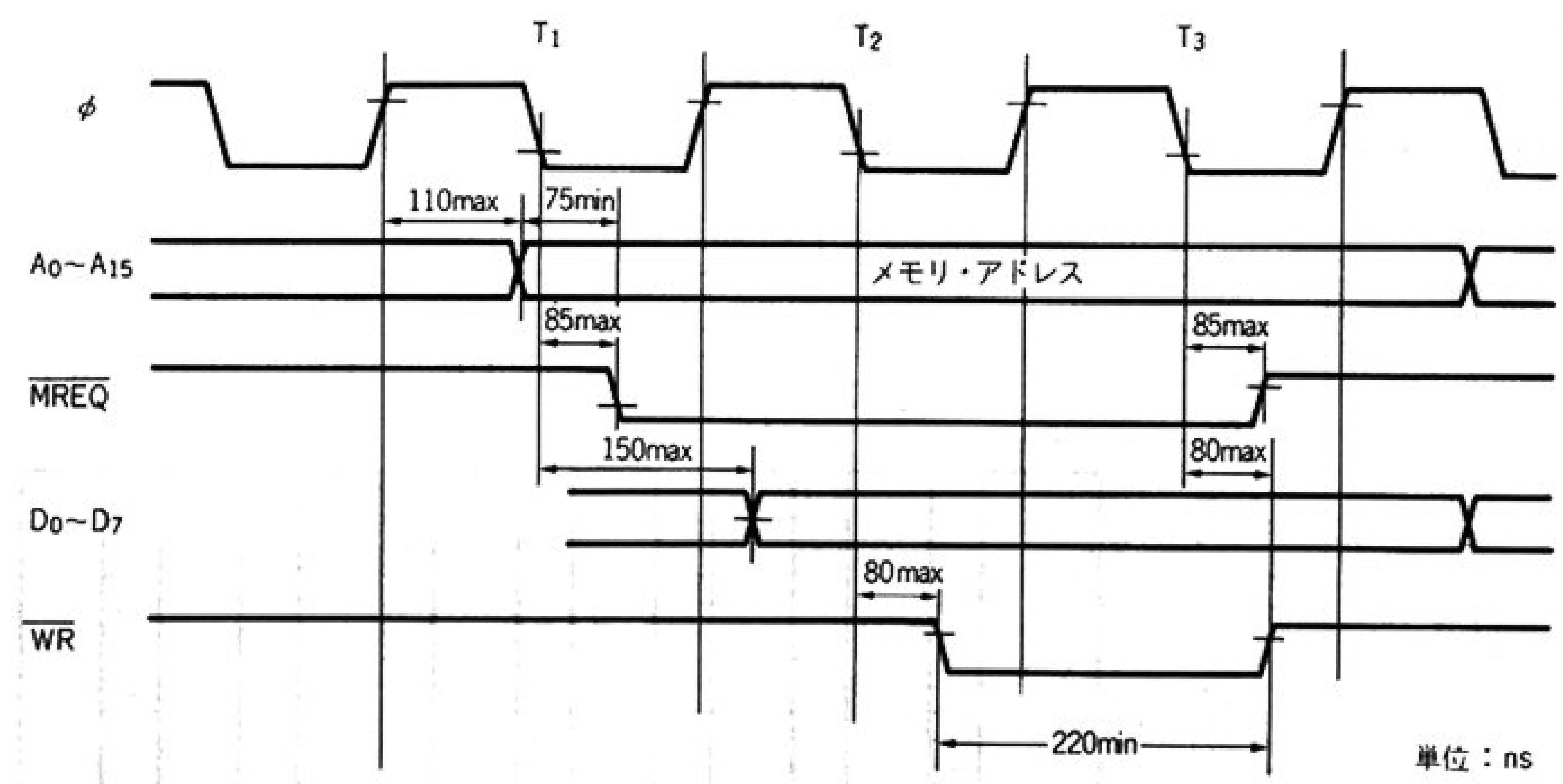
〈図8〉バースト・モードと分散型のリフレッシュ



〈図9〉
Z 80 CPUのM1サイクルのタイミング
(CPUのクロックは4 MHz)



〈図10〉
Z 80 A CPUのメモリ・リード・サイクルのタイミング
(CPUのクロックは4 MHz)



〈図11〉
Z 80 CPUのメモリ・ライト・サイクルのタイミング
(CPUのクロックは4 MHz)

レッシュを行います．

どちらの方法を用いても，リフレッシュ動作に要する時間のトータルは同じですが，バースト・モードの場合には，比較的長時間CPUの動作が停止させられるので，フロッピー・ディスクなどが使われるシステムでは，不都合を生じます．

したがって，DRAMのリフレッシュは分散型で行うのがふつうです．Z80 CPUはM1サイクルごとに各行のリフレッシュを行いますので，Z80 CPUのリフレッシュ機能を用いる場合には，分散型のリフレッシュを行うことができます．

Z80 CPU のメモリ・サイクル・タイミング

図9～図11にZ80A CPUを4MHzで動作させたときの，M1サイクル，メモリ・リード・サイクル，ならびにメモリ・ライト・サイクルのタイミングを示します．Z80 CPUとDRAMのインターフェースをとる場合，$\overline{\text{MREQ}}$により$\overline{\text{RAS}}$を起動しますので，これらの図には$\overline{\text{MREQ}}$が出てから，CPUがメモリ・データを読み取るまでの時間が示されています．

〈図12〉 64K DRAMインターフェースの構成

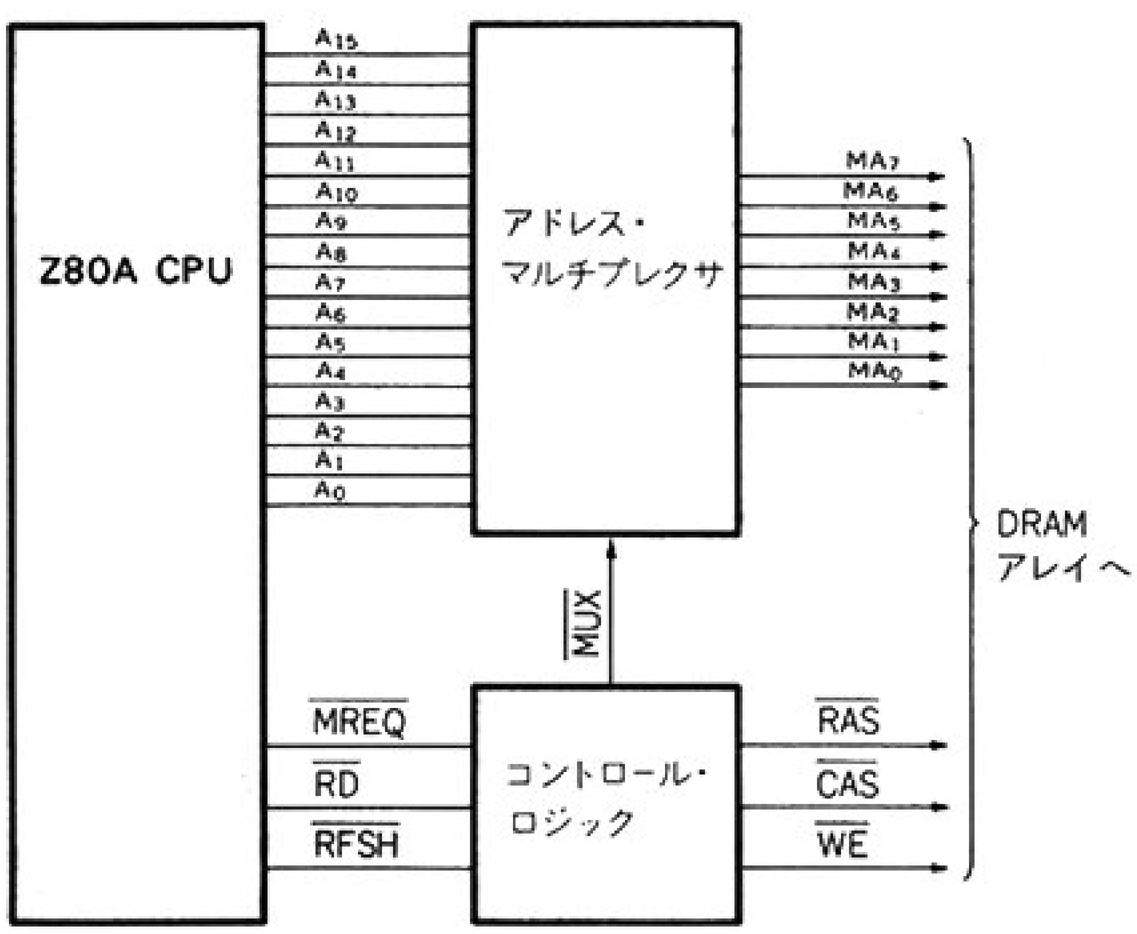


図9および図10からわかるように，Z80A CPUを最高速度の4MHzで動作させた場合に必要なDRAMのアクセス・タイムは，M1サイクルでは255ns，メモリ・リード・サイクルでは365nsです．

したがって，M1サイクルにおけるタイミングが最悪ケースとなりますので，$\overline{\text{RAS}}$からのアクセス・タイムが255nsより小さいDRAMを使う必要があります．

M1サイクルではOPコードのリードとリフレッシュ動作のために$\overline{\text{MREQ}}$が2度出力されますので，DRAMには$\overline{\text{RAS}}$が2回出力されます．DRAMには$\overline{\text{RAS}}$プリチャージ・タイムという規定があり，$\overline{\text{RAS}}$が"H"になってから再度$\overline{\text{RAS}}$が"L"になるまでの時間の最小値が決められています．

図9からわかるように，M1サイクルにおいて二つの$\overline{\text{MREQ}}$間の時間の最小値は110nsです．したがって，$\overline{\text{RAS}}$プリチャージ・タイムが110ns以下のDRAMを選択する必要があります．

図5のタイミング定数を見ると，$\overline{\text{RAS}}$からアクセス・タイムが255nsで，$\overline{\text{RAS}}$プリチャージ・タイムが110ns以下のDRAMは，HM4864AP-15ということになります．

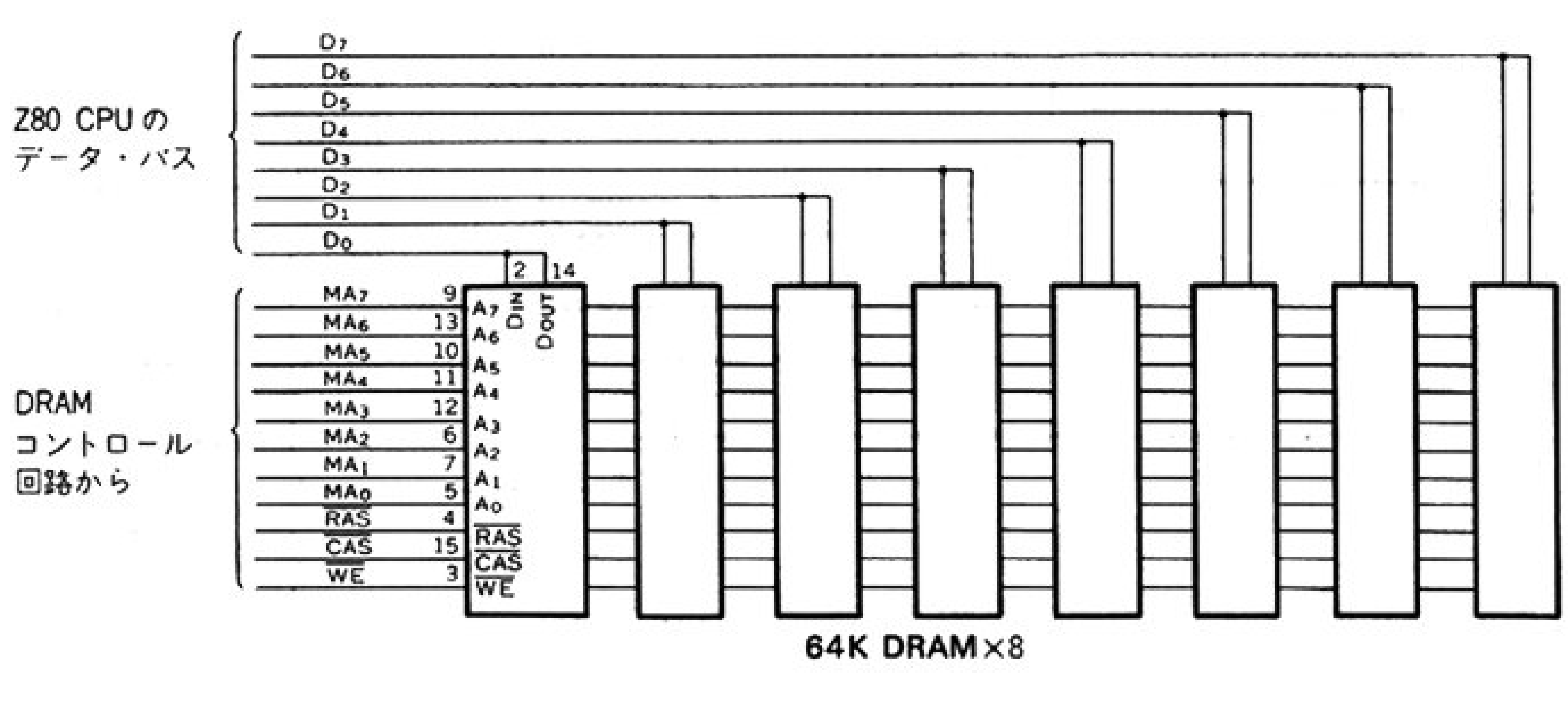
Z80 CPU と 64K DRAM のインターフェース

ここではZ80A CPUを4MHzで動作させる場合のDRAMとのインターフェース・ロジックを考えます．

図12に64K DRAMインターフェースの構成を示します．図中のコントロール・ロジックはCPUからの3本の制御線，すなわち$\overline{\text{MREQ}}$，$\overline{\text{RD}}$ならびに$\overline{\text{RFSH}}$を用いて，DRAMに接続される$\overline{\text{RAS}}$，$\overline{\text{CAS}}$，$\overline{\text{WE}}$信号を出力し，さらにアドレス・マルチプレクサを制御する$\overline{\text{MUX}}$信号を出力します．アドレス・マルチプレクサは$\overline{\text{MUX}}$信号により，CPUのメモリ・アドレスを行アドレス，および列アドレスにマルチプレックスします．

図13に64K DRAMを8個用いた64Kバイトのドラムアレイを示します．DRAMのアドレス・ライン，$\overline{\text{RAS}}$，$\overline{\text{CAS}}$ならびに$\overline{\text{WE}}$には図12のロジックから

〈図13〉
64KバイトDRAMアレイ
（DRAMとしてHM4864AP-15を用いる）



信号が供給されます．DRAMのデータ入力(D_{IN})およびデータ出力(D_{OUT})は互いに接続され，CPUのデータ・バスに接がっています．

図14に64 K DRAMのコントロール・ロジックを示します．このロジックでは$\overline{MUX}$信号，および$\overline{CAS}$を作るのにディレイ・ラインを用いています．CPUから$\overline{MREQ}$が出るとS 08を介して$\overline{RAS}$が出力されます．$\overline{RAS}$が出た後，二つのS 04およびディレイ・ラインにより，遅延時間30 ns経つと，$\overline{MUX}$が出力され，MA_7～MA_0上には列アドレスが出力されます．

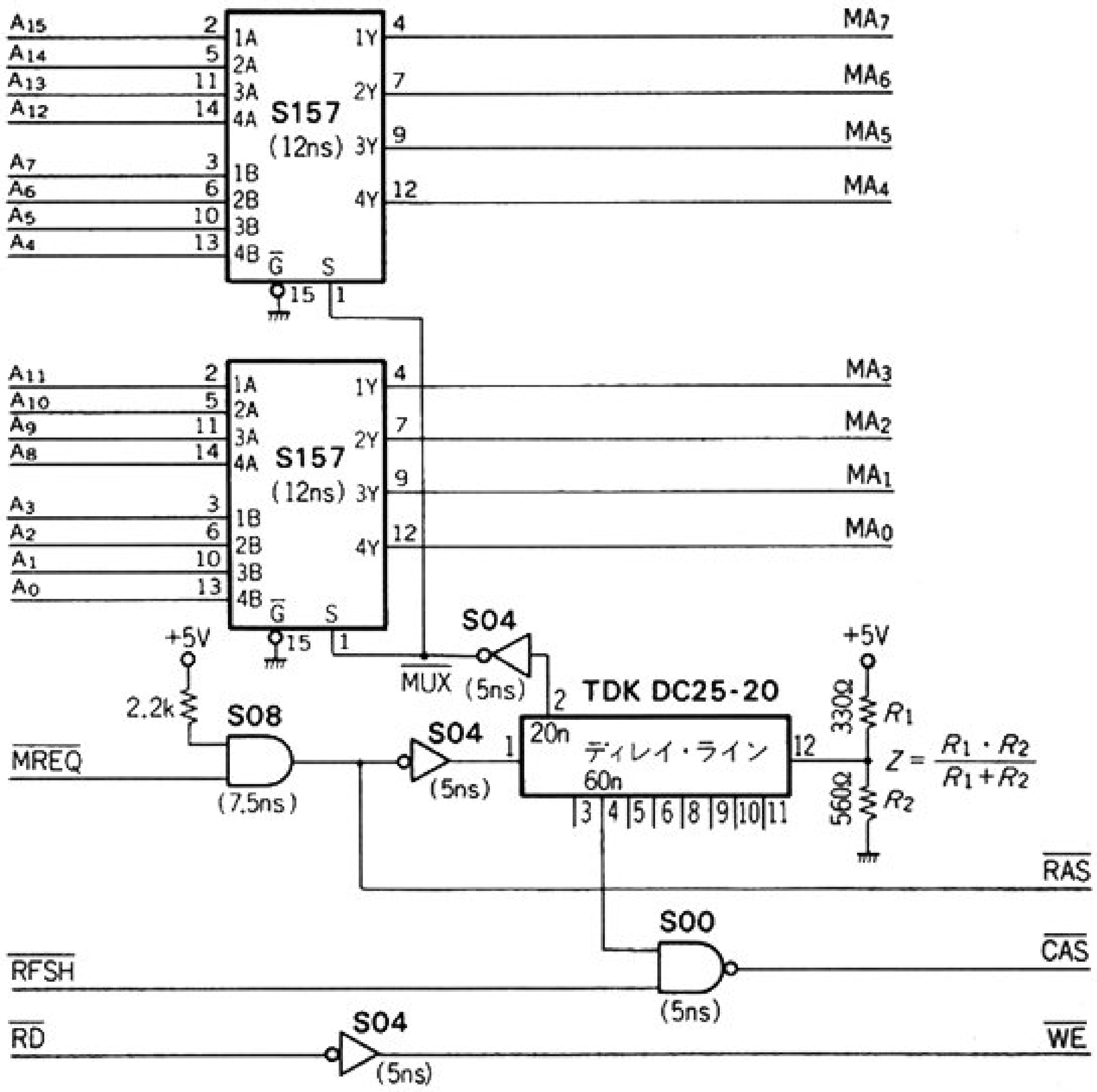
また，$\overline{MUX}$が出た後40 ns経つと，S 00を介して$\overline{CAS}$が出力されます．M 1サイクル中のリフレッシュのタイミングには$\overline{CAS}$が出ないように，このS 00には$\overline{RFSH}$信号が入力されています．

マルチプレクサS 157のセレクト入力(ピン①)は，$\overline{MREQ}$が“H”であると$\overline{MUX}$が“H”ですので，やはり“H”になります．したがってこのマルチプレクサは，行アドレスをB側の入力から出力します．これはZ 80 CPUがリフレッシュ・アドレスを下位7ビットに出力してくるからです．

DRAMの$\overline{WE}$信号として，CPUの$\overline{RD}$を反転したものを用いています．CPUの$\overline{WR}$信号はかなり遅く出力されるので，$\overline{WR}$が“L”になる前に，DRAMはリード・サイクルを実行します．図13のようにD_{IN}とD_{OUT}を接続した構成を用いていますので，CPUからのライト・データとメモリからのリード・データが衝突する危険性があるからです．

図15に図14のロジックのタイミングを示します．この図からわかるように，$\overline{RAS}$が出てから$\overline{CAS}$が出るまでの時間は70 nsです．図5のデータ・シートから，HM 4864 AP-15のt_{RCD}は25～75 nsです．したがって，アクセス・タイムとしては$\overline{RAS}$アクセス・タイムがそのまま適用され，150 nsとなります．

〈図14〉64 K DRAMコントロール・ロジック



〈図15〉図14のロジックのタイミング(単位：ns)

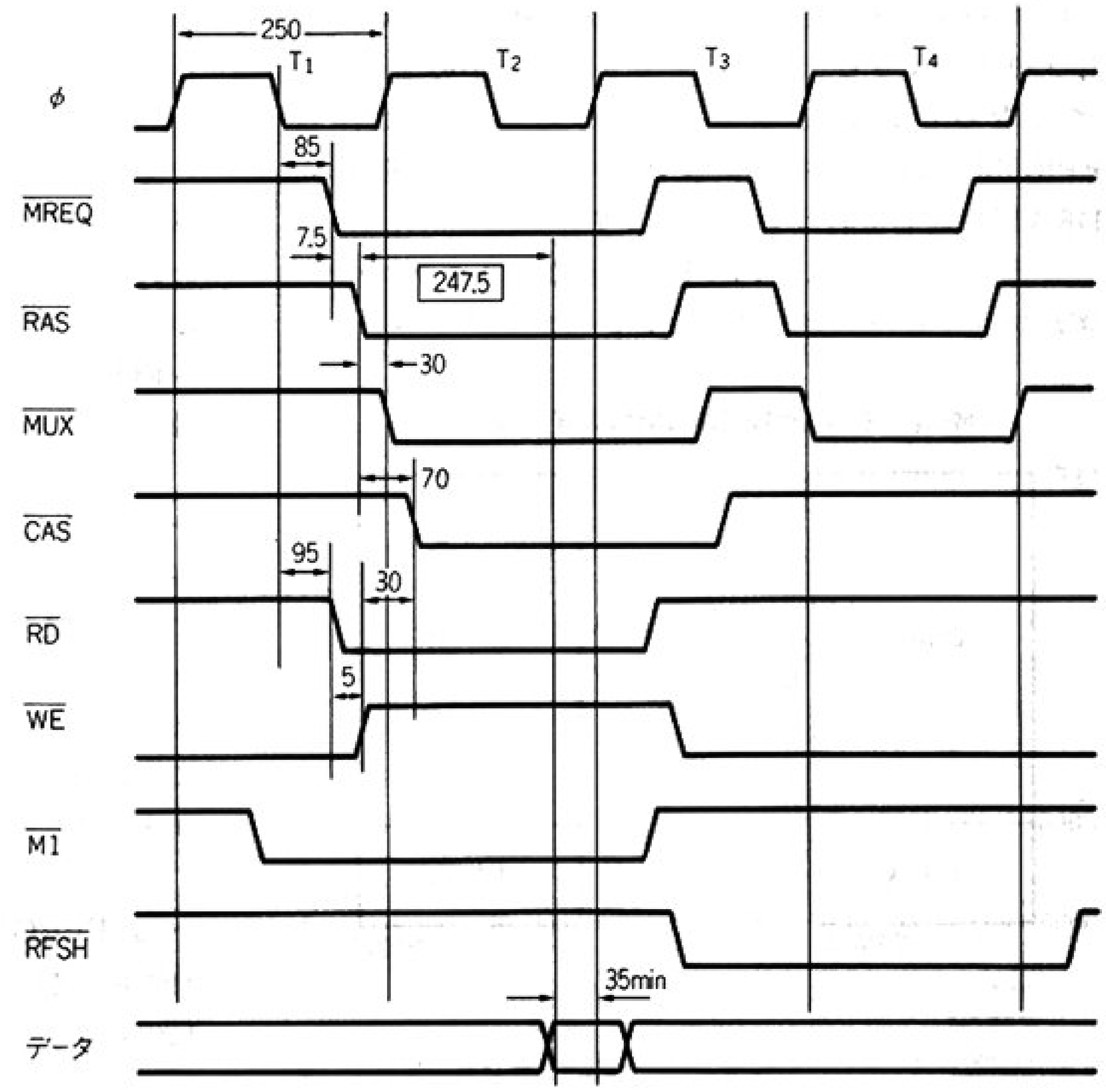
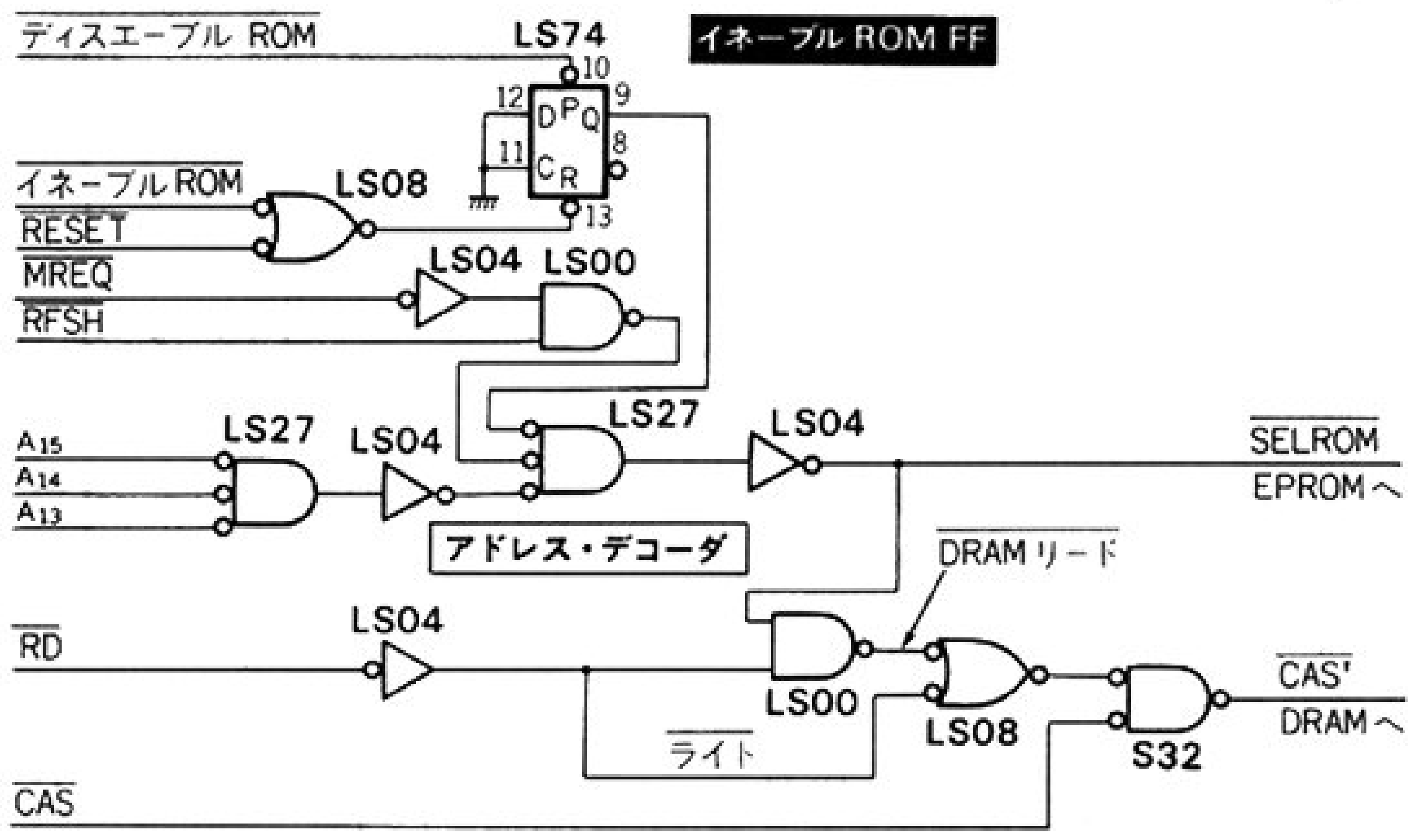


図15に示したように，$\overline{\text{RAS}}$が出てからCPUがデータを読み取るまでに247.5 nsありますので，図14のロジックは十分に動作可能であるということができます．

〈図17〉 EPROMとDRAMのスイッチングを行うロジック



EPROM と DRAM の スイッチング

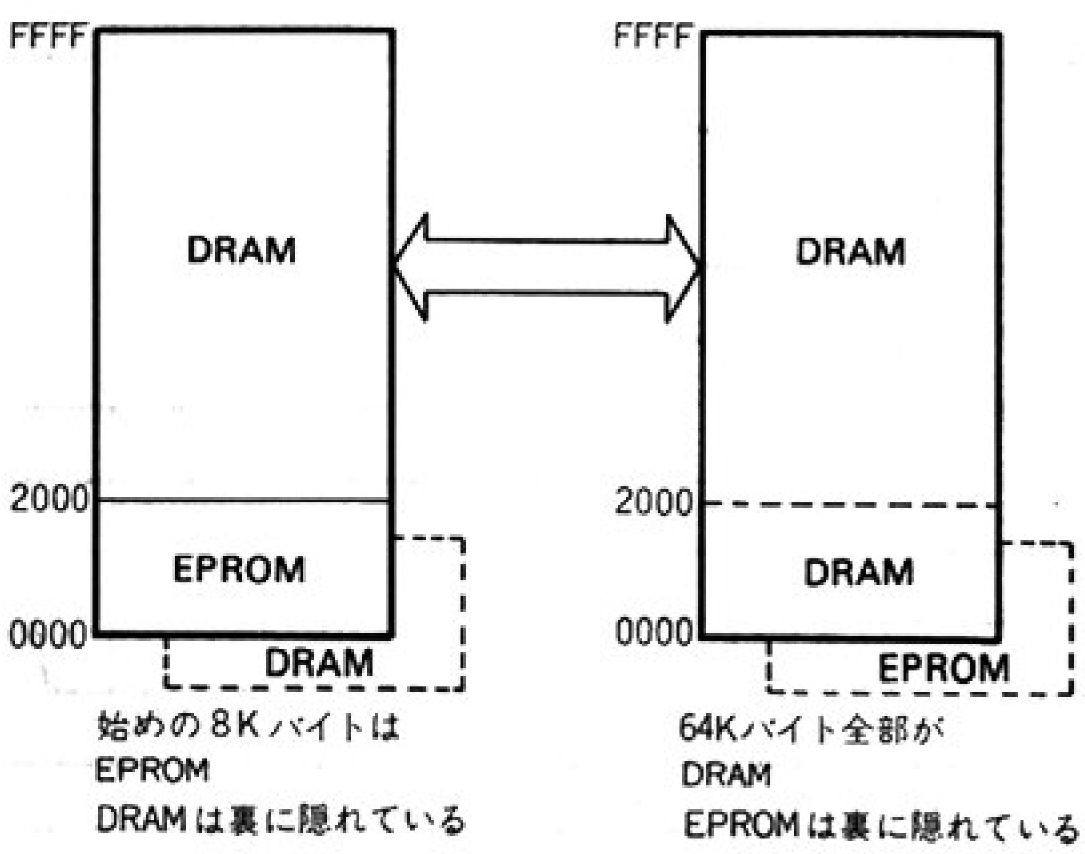
以上の説明ではDRAMが64 Kバイトのメモリ空間を全部占めていましたが，実際の応用ではこのようなことはありえず，必ずいくつかのEPROMが存在します．したがって64 Kバイトのメモリ空間の一部をEPROMのために削いてやる必要があります．EPROMにいくらかのアドレス空間を削いてやったとしても，その空間分だけDRAMを使わないのではもったいないので，両者をスイッチングする方法を考えてみます．

図16にEPROMとDRAMのスイッチングの概念を示します．左側の図では，始めの8 Kバイト(2000 H)はEPROMになっており，この空間部分のDRAMは裏に隠れて見えなくなっています．右側の図では，64 Kバイトのメモリ空間全体がDRAMになっており，EPROMは裏に隠れてしまっています．

CPUがリセットされたときには，図16左側のようなメモリ・マッピングにしておきます．このマッピングになっているときにはCPUのリード動作はEPROMに対して行い，ライト動作は裏に隠れたDRAMに対して行うことにします．

こうすることにより，EPROM中のプログラム・ローダがフロッピー・ディスクなどからプログラムをロードした後に，図16右側の図のようなマッピングにスイッチングすることにより，64 Kバイト全空間をDRAMとして走らせることができます．

〈図16〉 EPROMとDRAMのスイッチング



また64 Kバイト全空間がDRAMになっているときにも，2000 H番地以上にいるプログラムは，必要に応じてEPROMをめくり出し(図16の左側のマッピングにする)，EPROM中のサブルーチンなどを実行することもできます．このような使いかたをすると，実質的には72 Kバイトのメモリがあるかのようにふるまうことができます．

図17に図16の概念を実行に移すためのロジックを示します．この図中の$\overline{\text{CAS}}$は，図14のDRAMコントロール・ロジックが出力する$\overline{\text{CAS}}$と同じものです．このロジック中のイネーブルROMフリップフロップ(LS 74)は，リセット時にはクリヤされ，EPROMが現われている空間であることを示しています．また，このフリップフロップはI/O命令により，$\overline{\text{ディスエーブルROM}}$信号を出すとセットされ，$\overline{\text{イネーブルROM}}$信号を出すとクリヤされます．

イネーブルROMフリップフロップがクリヤされているときに，2000 H番地以下のアドレスがくるとEPROMをセレクトする$\overline{\text{SELROM}}$信号が出力されます．このときメモリ・リード動作であれば，$\overline{\text{CAS}}$はブロックされ，ライト動作であれば，$\overline{\text{CAS}}$はそのままパスし$\overline{\text{CAS}}'$としてDRAMに出力されます．なお，EPROMの$\overline{\text{OE}}$には$\overline{\text{RD}}$信号が接続されていなければなりません．

イネーブルROMフリップ・フロップがセットされている(EPROMをディスエーブルする)ときには，$\overline{\text{SELROM}}$信号は出力されなくなり，リード/ライト両動作に対して$\overline{\text{CAS}}'$が出力されます．

Appendix 4 Z80 CPU命令一覧表

以下に Z80 CPU の命令一覧を示します．Z80 CPU の持つ命令については，第2章でも詳しく述べましたが，ここではそれらを簡便に使用できるように一覧表にまとめてあります．ハンドアセンブル時などには便利でしょう．

なお，この表の中のオブジェクトコードのサンプル値は次のとおりです．

nn	EQU	584 H
d	EQU	5
n	EQU	20 H
e	EQU	30 H

（出典：Zilog Z80 CPU Programming Reference Card）

OBJ CODE	SOURCE STATEMENT		OPERATION
8E	ADC	A,(HL)	Add with Carry Operand to Acc.
DD8E05	ADC	A,(IX+d)	
FD8E05	ADC	A,(IY+d)	
8F	ADC	A,A	
88	ADC	A,B	
89	ADC	A,C	
8A	ADC	A,D	
8B	ADC	A,E	
8C	ADC	A,H	
8D	ADC	A,L	
CE20	ADC	A,n	
ED4A	ADC	HL,BC	Add with Carry Reg. Pair to HL
ED5A	ADC	HL,DE	
ED6A	ADC	HL,HL	
ED7A	ADC	HL,SP	
86	ADD	A,(HL)	Add Operand to Acc.
DD8605	ADD	A,(IX+d)	
FD8605	ADD	A,(IY+d)	
87	ADD	A,A	
80	ADD	A,B	
81	ADD	A,C	
82	ADD	A,D	
83	ADD	A,E	
84	ADD	A,H	
85	ADD	A,L	
C620	ADD	A,n	
09	ADD	HL,BC	Add Reg. Pair to HL
19	ADD	HL,DE	
29	ADD	HL,HL	
39	ADD	HL,SP	
DD09	ADD	IX,BC	Add Reg. Pair to IX
DD19	ADD	IX,DE	
DD29	ADD	IX,IX	
DD39	ADD	IX,SP	
FD09	ADD	IY,BC	Add Reg. Pair to IY
FD19	ADD	IY,DE	
FD29	ADD	IY,IY	
FD39	ADD	IY,SP	

OBJ CODE	SOURCE STATEMENT		OPERATION
A6	AND	(HL)	Logical 'AND' of Operand and Acc.
DDA605	AND	(IX+d)	
FDA605	AND	(IY+d)	
A7	AND	A	
A0	AND	B	
A1	AND	C	
A2	AND	D	
A3	AND	E	
A4	AND	H	
A5	AND	L	
E620	AND	n	
CB46	BIT	0,(HL)	Test Bit b of Location or Reg.
DDCB0546	BIT	0,(IX+d)	
FDCB0546	BIT	0,(IY+d)	
CB47	BIT	0,A	
CB40	BIT	0,B	
CB41	BIT	0,C	
CB42	BIT	0,D	
CB43	BIT	0,E	
CB44	BIT	0,H	
CB45	BIT	0,L	
CB4E	BIT	1,(HL)	
DDCB054E	BIT	1,(IX+d)	
FDCB054E	BIT	1,(IY+d)	
CB4F	BIT	1,A	
CB48	BIT	1,B	
CB49	BIT	1,C	
CB4A	BIT	1,D	
CB4B	BIT	1,E	
CB4C	BIT	1,H	
CB4D	BIT	1,L	
CB56	BIT	2,(HL)	
DDCB0556	BIT	2,(IX+d)	
FDCB0556	BIT	2,(IY+d)	
CB57	BIT	2,A	
CB50	BIT	2,B	
CB51	BIT	2,C	
CB52	BIT	2,D	
CB53	BIT	2,E	
CB54	BIT	2,H	
CB55	BIT	2,L	

OBJ CODE	SOURCE STATEMENT		OPERATION
CB5E	BIT	3,(HL)	Test Bit b Location or Reg.
DDCB055E	BIT	3,(IX+d)	
FDCB055E	BIT	3,(IY+d)	
CB5F	BIT	3,A	
CB58	BIT	3,B	
CB59	BIT	3,C	
CB5A	BIT	3,D	
CB5B	BIT	3,E	
CB5C	BIT	3,H	
CB5D	BIT	3,L	
CB66	BIT	4,(HL)	
DDCB0566	BIT	4,(IX+d)	
FDCB0566	BIT	4,(IY+d)	
CB67	BIT	4,A	
CB60	BIT	4,B	
CB61	BIT	4,C	
CB62	BIT	4,D	
CB63	BIT	4,E	
CB64	BIT	4,H	
CB65	BIT	4,L	
CB6E	BIT	5,(HL)	
DDCB056E	BIT	5,(IX+d)	
FDCB056E	BIT	5,(IY+d)	
CB6F	BIT	5,A	
CB68	BIT	5,B	
CB69	BIT	5,C	
CB6A	BIT	5,D	
CB6B	BIT	5,E	
CB6C	BIT	5,H	
CB6D	BIT	5,L	
CB76	BIT	6,(HL)	
DDCB0576	BIT	6,(IX+d)	
FDCB0576	BIT	6,(IY+d)	
CB77	BIT	6,A	
CB70	BIT	6,B	
CB71	BIT	6,C	
CB72	BIT	6,D	
CB73	BIT	6,E	
CB74	BIT	6,H	
CB75	BIT	6,L	
CB7E	BIT	7,(HL)	
DDCB057E	BIT	7,(IX+d)	
FDCB057E	BIT	7,(IY+d)	
CB7F	BIT	7,A	
CB78	BIT	7,B	
CB79	BIT	7,C	
CB7A	BIT	7,D	
CB7B	BIT	7,E	
CB7C	BIT	7,H	
CB7D	BIT	7,L	
DC8405	CALL	C,nn	Call Subroutine at Location nn if Condition True
FC8405	CALL	M,nn	
D48405	CALL	NC,nn	
C48405	CALL	NZ,nn	
F48405	CALL	P,nn	
EC8405	CALL	PE,nn	
E48405	CALL	PO,nn	
CC8405	CALL	Z,nn	
CD8405	CALL	nn	Unconditional Call to Subroutine at nn
3F	CCF		Complement Carry Flag
BE	CP	(HL)	Compare Operand with Acc.
DDBE05	CP	(IX+d)	
FDBE05	CP	(IY+d)	
BF	CP	A	
B8	CP	B	
B9	CP	C	
BA	CP	D	

OBJ CODE	SOURCE STATEMENT		OPERATION
BB	CP	E	Compare Operand with Acc.
BC	CP	H	
BD	CP	L	
FE20	CP	n	
EDA9	CPD		Compare Location (HL) and Acc. Decrement HL and BC
EDB9	CPDR		Compare Location (HL) and Acc. Decrement HL and BC, Repeat until BC = 0
EDA1	CPI		Compare Location (HL) and Acc., Increment HL and Decrement BC
EDB1	CPIR		Compare Location (HL) and Acc. Increment HL, Decrement BC, Repeat until BC = 0
2F	CPL		Complement Acc. (1's Comp)
27	DAA		Decimal Adjust Acc.
35	DEC	(HL)	Decrement Operand
DD3505	DEC	(IX+d)	
FD3505	DEC	(IY+d)	
3D	DEC	A	
05	DEC	B	
0B	DEC	BC	
0D	DEC	C	
15	DEC	D	
1B	DEC	DE	
1D	DEC	E	
25	DEC	H	
2B	DEC	HL	
DD2B	DEC	IX	
FD2B	DEC	IY	
2D	DEC	L	
3B	DEC	SP	
F3	DI		Disable Interrupts
102E	DJNZ	e	Decrement B and Jump Relative if B = 0
FB	EI		Enable Interrupts
E3	EX	(SP),HL	Exchange Location and (SP)
DDE3	EX	(SP),IX	
FDE3	EX	(SP),IY	
08	EX	AF,AF'	Exchange the Contents of AF and AF'
EB	EX	DE,HL	Exchange the Contents of DE and HL
D9	EXX		Exchange the Contents of BC,DE,HL with Contents of BC',DE',HL' Respectively
76	HALT		HALT (Wait for Interrupt or Reset)
ED46	IM	0	Set Interrupt Mode
ED56	IM	1	
ED5E	IM	2	

OBJ CODE	SOURCE STATEMENT		OPERATION
ED78	IN	A,(C)	Load Reg. with Input from Device (C)
ED40	IN	B,(C)	
ED48	IN	C,(C)	
ED50	IN	D,(C)	
ED58	IN	E,(C)	
ED60	IN	H,(C)	
ED68	IN	L,(C)	
34	INC	(HL)	Increment Operand
DD3405	INC	(IX+d)	
FD3405	INC	(IY+d)	
3C	INC	A	
04	INC	B	
03	INC	BC	
0C	INC	C	
14	INC	D	
13	INC	DE	
1C	INC	E	
24	INC	H	
23	INC	HL	
DD23	INC	IX	
FD23	INC	IY	
2C	INC	L	
33	INC	SP	
DB20	IN	A,(n)	Load Acc. with Input from Device n
EDAA	IND		Load Location (HL) with Input from Port (C), Decrement HL and B
EDBA	INDR		Load Location (HL) with Input from Port (C), Decrement HL and Decrement B, Repeat until B = 0
EDA2	INI		Load Location (HL) with Input from Port (C); Increment HL and Decrement B
EDB2	INIR		Load Location (HL) with Input from Port (C), Increment HL and Decrement B, Repeat until B = 0
C38405	JP	nn	Unconditional Jump to Location
E9	JP	(HL)	
DDE9	JP	(IX)	
FDE9	JP	(IY)	
DA8405	JP	C,nn	Jump to Location if Condition True
FA8405	JP	M,nn	
D28405	JP	NC,nn	
C28405	JP	NZ,nn	
F28405	JP	P,nn	
EA8405	JP	PE,nn	
E28405	JP	PO,nn	
CA8405	JP	Z,nn	
382E	JR	C,e	Jump Relative to PC+e if Condition True
302E	JR	NC,e	
202E	JR	NZ,e	
282E	JR	Z,e	
182E	JR	e	Unconditional Jump Relative to PC+e
02	LD	(BC),A	Load Source to Destination
12	LD	(DE),A	
77	LD	(HL),A	
70	LD	(HL),B	

OBJ CODE	SOURCE STATEMENT		OPERATION
71	LD	(HL),C	Load Source to Destination
72	LD	(HL),D	
73	LD	(HL),E	
74	LD	(HL),H	
75	LD	(HL),L	
3620	LD	(HL),n	
DD7705	LD	(IX+d),A	
DD7005	LD	(IX+d),B	
DD7105	LD	(IX+d),C	
DD7205	LD	(IX+d),D	
DD7305	LD	(IX+d),E	
DD7405	LD	(IX+d),H	
DD7505	LD	(IX+d),L	
DD360520	LD	(IX+d),n	
FD7705	LD	(IY+d),A	
FD7005	LD	(IY+d),B	
FD7105	LD	(IY+d),C	
FD7205	LD	(IY+d),D	
FD7305	LD	(IY+d),E	
FD7405	LD	(IY+d),H	
FD7505	LD	(IY+d),L	
FD360520	LD	(IY+d),n	
328405	LD	(nn),A	
ED438405	LD	(nn),BC	
ED538405	LD	(nn),DE	
228405	LD	(nn),HL	
DD228405	LD	(nn),IX	
FD228405	LD	(nn),IY	
ED738405	LD	(nn),SP	
0A	LD	A,(BC)	
1A	LD	A,(DE)	
7E	LD	A,(HL)	
DD7E05	LD	A,(IX+d)	
FD7E05	LD	A,(IY+d)	
3A8405	LD	A,(nn)	
7F	LD	A,A	
78	LD	A,B	
79	LD	A,C	
7A	LD	A,D	
7B	LD	A,E	
7C	LD	A,H	
ED57	LD	A,I	
7D	LD	A,L	
3E20	LD	A,n	
ED5F	LD	A,R	
46	LD	B,(HL)	
DD4605	LD	B,(IX+d)	
FD4605	LD	B,(IY+d)	
47	LD	B,A	
40	LD	B,B	
41	LD	B,C	
42	LD	B,D	
43	LD	B,E	
44	LD	B,H	
45	LD	B,L	
0620	LD	B,n	
ED4B8405	LD	BC,(nn)	
018405	LD	BC,nn	
4E	LD	C,(HL)	
DD4E05	LD	C,(IX+d)	
FD4E05	LD	C,(IY+d)	
4F	LD	C,A	
48	LD	C,B	
49	LD	C,C	
4A	LD	C,D	
4B	LD	C,E	
4C	LD	C,H	
4D	LD	C,L	
0E20	LD	C,n	
56	LD	D,(HL)	
DD5605	LD	D,(IX+d)	

OBJ CODE	SOURCE STATEMENT		OPERATION
FD5605	LD	D,(IY+d)	Load Source to Des-
57	LD	D,A	tination
50	LD	D,B	
51	LD	D,C	
52	LD	D,D	
53	LD	D,E	
54	LD	D,H	
55	LD	D,L	
1620	LD	D,n	
ED5B8405	LD	DE,(nn)	
118405	LD	DE,nn	
5E	LD	E,(HL)	
DD5E05	LD	E,(IX+d)	
FD5E05	LD	E,(IY+d)	
5F	LD	E,A	
58	LD	E,B	
59	LD	E,C	
5A	LD	E,D	
5B	LD	E,E	
5C	LD	E,H	
5D	LD	E,L	
1E20	LD	E,n	
66	LD	H,(HL)	
DD6605	LD	H,(IX+d)	
FD6605	LD	H,(IY+d)	
67	LD	H,A	
60	LD	H,B	
61	LD	H,C	
62	LD	H,D	
63	LD	H,E	
64	LD	H,H	
65	LD	H,L	
2620	LD	H,n	
2A8405	LD	HL,(nn)	
218405	LD	HL,nn	
ED47	LD	I,A	
DD2A8405	LD	IX,(nn)	
DD218405	LD	IX,nn	
FD2A8405	LD	IY,(nn)	
FD218405	LD	IY,nn	
6E	LD	L,(HL)	
DD6E05	LD	L,(IX+d)	
FD6E05	LD	L,(IY+d)	
6F	LD	L,A	
68	LD	L,B	
69	LD	L,C	
6A	LD	L,D	
6B	LD	L,E	
6C	LD	L,H	
6D	LD	L,L	
2E20	LD	L,n	
ED4F	LD	R,A	
ED7B8405	LD	SP,(nn)	
F9	LD	SP,HL	
DDF9	LD	SP,IX	
FDF9	LD	SP,IY	
318405	LD	SP,nn	
EDA8	LDD		Load Location (DE) with Location (HL), Decrement DE,HL and BC
EDB8	LDDR		Load Location (DE) with Location (HL), Repeat until BC = 0
EDA0	LDI		Load Location (DE) with Location (HL), Increment DE,HL, Decrement BC

OBJ CODE	SOURCE STATEMENT		OPERATION
EDB0	LDIR		Load Location (DE) with Location (HL), Increment DE,HL, Decrement BC and Repeat until BC = 0
ED44	NEG		Negate Acc. (2's Complement)
00	NOP		No Operation
B6	OR	(HL)	Logical "OR" of
DDB605	OR	(IX+d)	Operand and Acc.
FDB605	OR	(IY+d)	Logical "OR" of
B7	OR	A	Operand and Acc.
B0	OR	B	
B1	OR	C	
B2	OR	D	
B3	OR	E	
B4	OR	H	
B5	OR	L	
F620	OR	n	
ED8B	OTDR		Load Output Port (C) with Location (HL) Decrement HL and B, Repeat until B = 0
EDB3	OTIR		Load Output Port (C) with Location (HL), Increment HL, Decrement B, Repeat until B = 0
ED79	OUT	(C),A	Load Output Port (C)
ED41	OUT	(C),B	with Reg.
ED49	OUT	(C),C	
ED51	OUT	(C),D	
ED59	OUT	(C),E	
ED61	OUT	(C),H	
ED69	OUT	(C),L	
D320	OUT	(n),A	Load Output Port (n) with Acc.
EDAB	OUTD		Load Output Port (C) with Location (HL), Decrement HL and B
EDA3	OUTI		Load Output Port (C) with Location (HL), Increment HL and Decrement B
F1	POP	AF	Load Destination
C1	POP	BC	with Top of Stack
D1	POP	DE	
E1	POP	HL	
DDE1	POP	IX	
FDE1	POP	IY	
F5	PUSH	AF	Load Source to Stack
C5	PUSH	BC	
D5	PUSH	DE	
E5	PUSH	HL	
DDE5	PUSH	IX	
FDE5	PUSH	IY	
CB86	RES	0,(HL)	Reset Bit b of
DDCB0586	RES	0,(IX+d)	Operand
FDCB0586	RES	0,(IY+d)	
CB87	RES	0,A	
CB80	RES	0,B	
CB81	RES	0,C	
CB82	RES	0,D	
CB83	RES	0,E	
CB84	RES	0,H	
CB85	RES	0,L	

OBJ CODE	SOURCE STATEMENT		OPERATION
CB8E	RES	1,(HL)	Reset Bit b of Operand
DDCB058E	RES	1,(IX+d)	
FDCB058E	RES	1,(IY+d)	
CB8F	RES	1,A	
CB88	RES	1,B	
CB89	RES	1,C	
CB8A	RES	1,D	
CB8B	RES	1,E	
CB8C	RES	1,H	
CB8D	RES	1,L	
CB96	RES	2,(HL)	
DDCB0596	RES	2,(IX+d)	
FDCB0596	RES	2,(IY+d)	
CB97	RES	2,A	
CB90	RES	2,B	
CB91	RES	2,C	
CB92	RES	2,D	
CB93	RES	2,E	
CB94	RES	2,H	
CB95	RES	2,L	
CB9E	RES	3,(HL)	
DDCB059E	RES	3,(IX+d)	
FDCB059E	RES	3,(IY+d)	
CB9F	RES	3,A	
CB98	RES	3,B	
CB99	RES	3,C	
CB9A	RES	3,D	
CB9B	RES	3,E	
CB9C	RES	3,H	
CB9D	RES	3,L	
CBA6	RES	4,(HL)	
DDCB05A6	RES	4,(IX+d)	
FDCB05A6	RES	4,(IY+d)	
CBA7	RES	4,A	
CBA0	RES	4,B	
CBA1	RES	4,C	
CBA2	RES	4,D	
DBA3	RES	4,E	
CBA4	RES	4,H	
CBA5	RES	4,L	
CBAE	RES	5,(HL)	
DDCB05AE	RES	5,(IX+d)	
FDCB05AE	RES	5,(IY+d)	
CBAF	RES	5,A	
CBA8	RES	5,B	
CBA9	RES	5,C	
CBAA	RES	5,D	
CBAB	RES	5,E	
CBAC	RES	5,H	
CBAD	RES	5,L	
CBB6	RES	6,(HL)	
DDCB05B6	RES	6,(IX+d)	
FDCB05B6	RES	6,(IY+d)	
CBB7	RES	6,A	
CBB0	RES	6,B	
CBB1	RES	6,C	
CBB2	RES	6,D	
CBB3	RES	6,E	
CBB4	RES	6,H	
CBB5	RES	6,L	
CBBE	RES	7,(HL)	
DDCB05BE	RES	7,(IX+d)	
FDCB05BE	RES	7,(IY+d)	
CBBF	RES	7,A	
CBB8	RES	7,B	
CBB9	RES	7,C	
CBBA	RES	7,D	
CBBB	RES	7,E	
CBBC	RES	7,H	
CBBD	RES	7,L	

OBJ CODE	SOURCE STATEMENT		OPERATION
C9	RET		Return from Subroutine
D8	RET	C	Return from Subroutine if Condition True
F8	RET	M	
D0	RET	NC	
C0	RET	NZ	
F0	RET	P	
E8	RET	PE	
E0	RET	PO	
C8	RET	Z	
ED4D	RETI		Return from Interrupt
ED45	RETN		Return from Non-Maskable Interrupt
CB16	RL	(HL)	Rotate Left Through Carry
DDCB0516	RL	(IX+d)	
FDCB0516	RL	(IY+d)	
CB17	RL	A	
CB10	RL	B	
CB11	RL	C	
CB12	RL	D	
CB13	RL	E	
CB14	RL	H	
CB15	RL	L	
17	RLA		Rotate Left Acc. Through Carry
CB06	RLC	(HL)	Rotate Left Circular
DDCB0506	RLC	(IX+d)	
FDCB0506	RLC	(IY+d)	
CB07	RLC	A	
CB00	RLC	B	
CB01	RLC	C	
CB02	RLC	D	
CB03	RLC	E	
CB04	RLC	H	
CB05	RLC	L	
07	RLCA		Rotate Left Circular Acc.
ED6F	RLD		Rotate Digit Left and Right between Acc. and Location (HL)
CB1E	RR	(HL)	Rotate Right Through Carry
DDCB051E	RR	(IX+d)	
FDCB051E	RR	(IY+d)	
CB1F	RR	A	
CB18	RR	B	
CB19	RR	C	
CB1A	RR	D	
CB1B	RR	E	
CB1C	RR	H	
CB1D	RR	L	
1F	RRA		Rotate Right Acc. Through Carry
CB0E	RRC	(HL)	Rotate Right Circular
DDCB050E	RRC	(IX+d)	
FDCB050E	RRC	(IY+d)	
CB0F	RRC	A	
CB08	RRC	B	Rotate Right Circular
CB09	RRC	C	
CB0A	RRC	D	
CB0B	RRC	E	
CB0C	RRC	H	
CB0D	RRC	L	
0F	RRCA		Rotate Right Circular Acc.

OBJ CODE	SOURCE STATEMENT		OPERATION
ED67	RRD		Rotate Digit Right and Left Between Acc. and Location (HL)
C7	RST	00H	Restart to Location
CF	RST	08H	
D7	RST	10H	
DF	RST	18H	
E7	RST	20H	
EF	RST	28H	
F7	RST	30H	
FF	RST	38H	
DE20	SBC	A,n	Subtract Operand from Acc. with Carry
9E	SBC	A,(HL)	
DD9E05	SBC	A,(IX+d)	
FD9E05	SBC	A,(IY+d)	
9F	SBC	A,A	
98	SBC	A,B	
99	SBC	A,C	
9A	SBC	A,D	
9B	SBC	A,E	
9C	SBC	A,H	
9D	SBC	A,L	
ED42	SBC	HL,BC	
ED52	SBC	HL,DE	
ED62	SBC	HL,HL	
ED72	SBC	HL,SP	
37	SCF		Set Carry Flag (C = 1)
CBC6	SET	0,(HL)	Set Bit b of Location
DDCB05C6	SET	0,(IX+d)	
FDCB05C6	SET	0,(IY+d)	
CBC7	SET	0,A	
CBC0	SET	0,B	
CBC1	SET	0,C	
CBC2	SET	0,D	
CBC3	SET	0,E	
CBC4	SET	0,H	
CBC5	SET	0,L	
CBCE	SET	1,(HL)	
DDCB05CE	SET	1,(IX+d)	
FDCB05CE	SET	1,(IY+d)	
CBCF	SET	1,A	
CBC8	SET	1,B	
CBC9	SET	1,C	
CBCA	SET	1,D	
CBCB	SET	1,E	
CBCC	SET	1,H	
CBCD	SET	1,L	
CBD6	SET	2,(HL)	
DDCB05D6	SET	2,(IX+d)	
FDCB05D6	SET	2,(IY+d)	
CBD7	SET	2,A	
CBD0	SET	2,B	
CBD1	SET	2,C	
CBD2	SET	2,D	
CBD3	SET	2,E	
CBD4	SET	2,H	
CBD5	SET	2,L	
CBD8	SET	3,B	
CBDE	SET	3,(HL)	
DDCB05DE	SET	3,(IX+d)	
FDCB05DE	SET	3,(IY+d)	
CBDF	SET	3,A	
CBD9	SET	3,C	
CBDA	SET	3,D	
CBDB	SET	3,E	
CBDC	SET	3,H	
CBDD	SET	3,L	
CBE6	SET	4,(HL)	
DDCB05E6	SET	4,(IX+d)	

OBJ CODE	SOURCE STATEMENT		OPERATION
FDCB05E6	SET	4,(IY+d)	Set Bit b of Location
CBE7	SET	4,A	
CBE0	SET	4,B	
CBE1	SET	4,C	
CBE2	SET	4,D	
CBE3	SET	4,E	
CBE4	SET	4,H	
CBE5	SET	4,L	
CBEE	SET	5,(HL)	
DDCB05EE	SET	5,(IX+d)	
FDCB05EE	SET	5,(IY+d)	
CBEF	SET	5,A	
CBE8	SET	5,B	
CBE9	SET	5,C	
CBEA	SET	5,D	
CBEB	SET	5,E	
CBEC	SET	5,H	
CBED	SET	5,L	
CBF6	SET	6,(HL)	
DDCB05F6	SET	6,(IX+d)	
FDCB05F6	SET	6,(IY+d)	
CBF7	SET	6,A	
CBF0	SET	6,B	
CBF1	SET	6,C	
CBF2	SET	6,D	
CBF3	SET	6,E	
CBF4	SET	6,H	
CBF5	SET	6,L	
CBFE	SET	7,(HL)	
DDCB05FE	SET	7,(IX+d)	
FDCB05FE	SET	7,(IY+d)	
CBFF	SET	7,A	
CBF8	SET	7,B	
CBF9	SET	7,C	
CBFA	SET	7,D	
CBFB	SET	7,E	
CBFC	SET	7,H	
CBFD	SET	7,L	
CB26	SLA	(HL)	Shift Operand Left Arithmetic
DDCB0526	SLA	(IX+d)	
FDCB0526	SLA	(IY+d)	
CB27	SLA	A	
CB20	SLA	B	
CB21	SLA	C	
CB22	SLA	D	
CB23	SLA	E	
CB24	SLA	H	
CB25	SLA	L	
CB2E	SRA	(HL)	Shift Operand Right Arithmetic
DDCB052E	SRA	(IX+d)	
FDCB052E	SRA	(IY+d)	
CB2F	SRA	A	
CB28	SRA	B	
CB29	SRA	C	
CB2A	SRA	D	
CB2B	SRA	E	
CB2C	SRA	H	
CB2D	SRA	L	
CB3E	SRL	(HL)	Shift Operand Right Logical
DDCB053E	SRL	(IX+d)	
FDCB053E	SRL	(IY+d)	
CB3F	SRL	A	
CB38	SRL	B	
CB39	SRL	C	
CB3A	SRL	D	
CB3B	SRL	E	
CB3C	SRL	H	
CB3D	SRL	L	

OBJ CODE	SOURCE STATEMENT		OPERATION
96	SUB	(HL)	Subtract Operand from Acc.
DD9605	SUB	(IX+d)	
FD9605	SUB	(IY+d)	
97	SUB	A	
90	SUB	B	
91	SUB	C	
92	SUB	D	
93	SUB	E	
94	SUB	H	
95	SUB	L	
D620	SUB	n	

OBJ CODE	SOURCE STATEMENT		OPERATION
AE	XOR	(HL)	Exclusive "OR" Operand and Acc.
DDAE05	XOR	(IX+d)	
FDAE05	XOR	(IY+d)	
AF	XOR	A	
A8	XOR	B	
A9	XOR	C	
AA	XOR	D	
AB	XOR	E	
AC	XOR	H	
AD	XOR	L	
EE20	XOR	n	

■参考資料■

本書の執筆に当たり，下記のマニュアル，資料を参照させていただきました．

・Z80-CPU　Technical Manual
・Z80-PIO　Technical Manual
・Z80-CTC　Technical Manual
・Z80-SIO　Technical Manual
・Z80-DMA　Technical Manual
・Z80　Assembly Language Programming Manual
・80-CPU　Programming Reference Card
・Z80 Family Program Interrupt Structure
・Application Note(Interfacing 16 Pin Dynamic RAMs to the Z80A Microprocessor)

(以上すべてZilog社発行資料)

著者略歴

額田忠之

1943年　東京に生まれる
1967年　早稲田大学理工学部電気工学科卒
現　在　コンピュータ企業にてパソコン，ターミナル等の開発に従事

Z80ファミリ・ハンドブック

1985年1月10日　初版発行
1995年7月1日　第18版発行

©額田忠之 1985

著　者　額田忠之
発行人　蒲生良治
発行所　CQ出版株式会社
東京都豊島区巣鴨1—14—2　(〒170)
電話　出版部 03-5395-2122　営業部 03-5395-2141
振替　00100-7-10665

(定価は表四に表示してあります)

乱丁，落丁本はお取り替えします　印刷・製本　図印刷株式会社　Printed in Japan

別冊 インターフェース　　好評発売中

Visual BASIC for Windowsの本格的応用



Windows DLLの利用からカスタム・コントロールの作成まで

B5判　144頁(3.5インチ2DD FD付き)　定価1,800円

Visual BASICは，従来のC言語とSDKによるWindowsプログラミングとは比較にならないくらい効率的な開発環境を提供してくれます．本書では，Visual BASICによって本格的なWindowsアプリケーションを作成する場合に必要となる，Windows DLLの呼び出し方法，アプリケーション間の連動処理，専用DLLの作成，RS-232-C制御用カスタム・コントロールのプログラミングなどについて，具体的なサンプル・プログラムを示して解説しました．

第1章 Visual BASICによるプログラミング
テキスト系ファイルの連続印刷／印刷方法指定機能／印刷前のイメージ表示機能／ほか

第2章 Windows DLLの利用/作成とVBXの活用
アプリケーションの機能分担／タスク・リストの検索方法／WindowsのAPIについて／アプリケーション間連動／KANAZOU2.DLL／FGE-TMES. VBXの機能と内部仕様

Appendix VBXを使う
フリーウェアVBX／シェアウェアVBX／市販VBX

第3章 カスタム・コントロールの作成
Windowsのシリアル通信の概要／シリアル通信制御関数セットを作る／シリアル通信制御カスタム・コントロールを作る／簡単なTTYアプリケーションを作る

IFセレクション　　好評発売中

MS-DOSディスク管理技法
中島信行 著　B5判　208頁　5″2HD FD付　定価2,500円
ブロック・デバイス・ドライバ作成からディスク管理メカニズムの解析まで

Cによるメモリ管理技法
中島信行 著　B5判　224頁　5″2HD FD付　定価2,500円
MS-DOS上でTSRプログラム&デバイス・ドライバを作成する

MS-DOS完全活用法
中島信行 著　B5判　224頁　5″2HD FD付　定価2,500円
DOS起動原理からユーティリティ作成まで

オブジェクト指向のすべて
小暮裕明 著　B5判　176頁　定価1,650円
'90年代プログラマの必須知識 OOPSをマスタする

MS-DOSメモリ管理ソフト技法
中島信行 著　B5判　224頁　定価1,650円
メモリ常駐ソフト&拡張メモリ活用プログラミング

DSPを使いこなす
インターフェース編集部 編　B5判　256頁　定価1,900円
ディジタル信号処理の理論からシステムの実現まで

MS-DOSプログラマーズ・バイブル
阿部英志 著　B5判　384頁　定価2,500円
内部構造とシステム・コールとデバイス・ドライバと…

データ管理の技法
本田道夫/藤 健児 共著　B5判　260頁　5″2HD FD付　定価2,900円
ISAMライブラリとその応用

CQ出版社　〒170 東京都豊島区巣鴨1-14-2　営業部☎03-5395-2141　振替00100-7-10665



CQ出版社 定価1,960円（本体1,903円） ISBN4-7898-3470-0 C3055 P1960E